OPTIMUM DIGITAL CONTROL SYNTHESIS

DON J. OGDEN

OPTIMUM DIGITAL CONTROL SYNTHESIS

* * * * * *

Captain Don J. Ogden USMC

OPTIMUM DIGITAL CONTROL SYNTHESIS

By

Don J. Ogden

Captain, United States Marine Corps

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1 9 6 5

OPTIMUM DIGITAL CONTROL SYNTHESIS

by

Captain Don J. Ogden USMC

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

from the

United States Naval Postgraduate School

ABSTRACT

Sampled-data control systems are coming of age. Many methods have been formulated by which both continuous and sampled-data control systems may be optimized. Here one method of optimum control design is presented using the approach of dynamic programming. A program to generate the state transition matrices from the system dynamics was developed. A program for minimizing rather general cost functions was written and examples are presented utilizing the results.

TABLE OF CONTENTS

Section	Title	Page
1.	Introduction	1
2.	PROGRAM OPCON1	6
3.	PROGRAM OPCON2	23
4.	Example Problem	45
5.	Bibliography	50
6.	Appendix I	51
7.	Appendix II	55
8.	Appendix III	62
9.	Appendix IV	69

LIST OF ILLUSTRATIONS

Figure		Page
I-1	Block Diagram of an Open Loop Control System	2
I-2	Flow Chart for PROGRAM PHIDEL	4
I-3	Multivariable Digital Control System	5
II-1	Phase Plane for Example Problem	11
II-2	Flow Chart of PROGRAM OPCON1	13-16
II-3	Flow Chart for Subroutine PROD(A,B,C,L,M,N)	17
II-4	Flow Chart for Subroutine SUM(A,B,C,N)	18
II-5	Flow Chart for Subroutine TRANSQ(A,B,N)	19
II-6	Flow Chart for Subroutine TRANCOL(A,B,N)	20
II-7	Flow Chart for Subroutine Money (MON,E,Q,N)	20
II-8	Flow Chart for Subroutine PPSI(B,C,D,E,F,N)	21
II-9	Flow Chart for Subroutine PP(A,B,C,N)	22
II-10	Flow Chart for Subroutine FF(A,B,C,D,N)	22
III-1	Flow Chart for PROGRAM OPCON2	29-35
III-2	Flow Chart for Subroutine ATRAN(AT,P,PHI,DEL,R,N)	36
III-3	Flow Chart for Subroutine PPSI(PSI,PHI,DEL,AT,N)	37
III-4	Flow Chart for Subroutine COST(DOL,Y,Q,R,Z,N)	37
III-5	Flow Chart for Subroutine PP(P,PSI,P1,Q,AT,R,N)	38-39
III-6	Phase Plane for Example Problem	43
IV-1	Block Diagram of the Open Loop Control System	45
IV-2	Plot of Control versus Time for the Four Cases	48
IV-3	Phase Plane Plot of the Four Cases	49

CHAPTER I

INTRODUCTION

In general, any linear control system may be expressed by the following state equation:

$$\dot{\underline{x}} = F\underline{x} + D\underline{u} \tag{1}$$

where x_i represents the system variables and u_i are the control variables. F and D are nxn and nxm matrices defining the dynamical relationship of the states and controls. We will sample the states at discrete intervals t_k and at a sampling rate with period T. A recursive relationship from the solution of the above differential equation may be established:

$$\underline{x}(k+1) = e^{FT}\underline{x}(k) + \int_0^T e^{F(T-\tau)} D\underline{u}(k)d\tau \tag{2}$$

where

$$e^{FT} = I + FT + \frac{F^2\ T^2}{2!} + \cdot\ \cdot\ \cdot + \frac{F^n\ T^n}{n!} \tag{3}$$

Equation (2) may be written:

$$\underline{x}(k+1) = \phi\underline{x}(k) + \Delta\underline{u}(k) \tag{4}$$

where

$$\phi = e^{FT} \tag{5}$$

and

$$\Delta = \int_0^T e^{F(T-\tau)} Dd\tau \tag{6}$$

As an example of the above equations, the open-loop system of Figure I-1 will be considered.

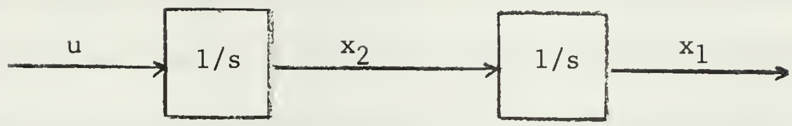

Figure I-1: Block diagram of an open loop control system.

By inspection, the following state equation defines the example system:

$$\underline{\dot{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underline{u} \tag{7}$$

The ϕ matrix as defined by (5) has only two non-zero terms in the Taylor expansion for the exponential function:

$$\phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} T \tag{8}$$

or

$$\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{9}$$

In a like manner, the convolution integral of (6) may be solved for Δ :

$$\Delta = \int_o^T \begin{bmatrix} 1 & T-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\tau \tag{10}$$

giving

$$\Delta = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \tag{11}$$

The system difference equation may now be written as a function of the sampling period T:

$$\underline{x}(k+1) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \underline{u}(k) \tag{12}$$

The calculation of the transition matrix, ϕ, and the external forcing function's impulse response matrix, Δ, from the matrices F and D,

of the process dynamics was accomplished with the digital computer. This program required an input of F and the sampling rate T. The amount of accuracy desired may be varied by the user by inputing the least significant figure desired. This test point for accuracy results from the fact that the sum of all terms in a Taylor series following the test term is less in magnitude than the test term. Figure I-2 is the flow chart of the program. The program has been incorporated into other programs written and presented in this paper.

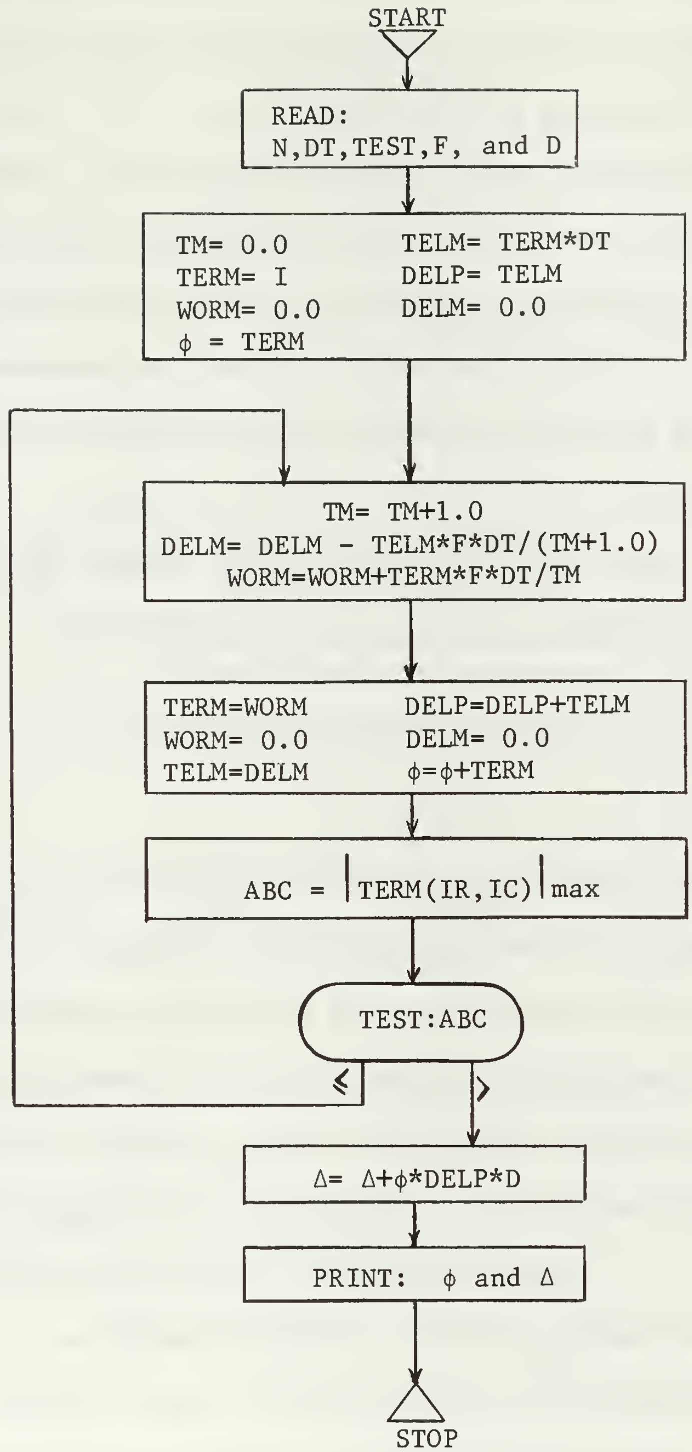

Figure I-2: Flow Chart for PROGRAM PHIDEL

The optimal control of a system depends upon a valid determination of an optimal control policy which will maximize or minimize a performance criteria. This control policy may be generated from criteria involving time, system state variables, noise, control effort, etc. In some cases, this control policy may be generated by active networks or filters incorporated within the system, or a digital computer may be used to generate the control. To accomplish this, all dynamics of the system must be fed back to the digital computer as indicated in Figure I-3.

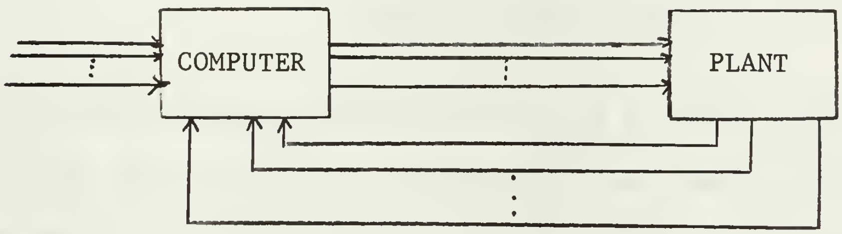

Figure I-3: Multivariable Digital Control System

A general approach to the optimization of any dynamic system was developed through the efforts of R. E. Bellman and has been called "dynamic programming". As an example of this method, in a missile problem, one might choose a proper cost function (terminal CEP) and minimize this cost throughout the trajectory. The optimum trajectory, according to the chosen cost function, would be computed.

Dynamic programming appears to be an excellent method for determining an optimal control process because of its adaptability to use on a digital computer. Throughout the remainder of this paper, this technique will be utilized in formulating general programs to synthesize optimum controllers.

CHAPTER II

PROGRAM OPCON1

Again let us consider the system:

$$\dot{\underline{x}} = F\underline{x} + \underline{D}\ u \tag{1'}$$

whose discrete solution is

$$\underline{x}(k+1) = \phi\underline{x}(k) + \underline{\Delta}u(k) \tag{4'}$$

where we now take the case where u(k) is a scalar control and $\underline{\Delta}$ is a column vector of system impulse responses due to control inputs. The recursion formulas will be derived for this system minimizing the sum of the quadratic terms as described by the matrix Q over an N stage process. The Q matrix is specified by the requirements of the user and the application of solid engineering judgment. This cost function may be written:

$$J(N) = \text{minimum} \sum_{k=1}^{N} \underline{x}^T(k)Q\underline{x}(k) \tag{13}$$

By starting with the last stage first, equation 13 (7) becomes:

$$J(N) = \text{minimum} \left\{ \underline{x}^T(N)Q\underline{x}(N) + \sum_{k=1}^{N-1} \underline{x}^T(k)Q\underline{x}(k) \right\}$$

or

$$J(N) = \text{minimum} \left\{ \underline{x}^T(N)Q\underline{x}(N) \right\} + J(N-1) \tag{14}$$

It will now be noted that only the $\underline{x}^T(N)Q\underline{x}(N)$ term contains the control policy u(N-1); therefore, to minimize this function over the last stage, we need only to find:

$$\frac{\partial(\underline{x}^T(N)Q\underline{x}(N))}{\partial\ u(N-1)} = 0 \tag{15}$$

By substitution of equation (4) into $\underline{x}^T(N)Q\underline{x}(N)$:

$$\underline{x}^T(N)Q\underline{x}(N) = \left[\phi\underline{x}(N-1)+\underline{\Delta}u(N-1)\right]^T Q \left[\phi\underline{x}(N-1)+\underline{\Delta}u(N-1)\right] \tag{16}$$

and

$$\frac{\partial(\underline{x}^T(N)Q\underline{\Delta}(N))}{\partial\ \underline{u}(N-1)} = 2\underline{\Delta}^T Q\phi\underline{x}(N-1) + 2\underline{\Delta}^T Q\underline{\Delta}\ u(N-1)=0 \tag{17}$$

or

$$\therefore\ u(N-1) = -\frac{\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\underline{x}(N-1) \tag{18}$$

Equation (18) now represents the optimum control policy u(N-1). It will be noted that we are dividing by a multiplication of three matrices; this is a valid operation because a row matrix times a square matrix times a column matrix will always yield a scalar number if the multiplication is valid. The multiplication is valid if the dimensions of the respective matrices are proper; and in this case, it will always be valid.

By proceeding back one more stage, the control policy u(N-2) may be optimized.

$$J(N) = \text{minimum}\left\{\underline{x}^T(N)Q\underline{x}(N)+\underline{x}^T(N-1)Q\underline{x}(N-1)\right\} + J(N-2) \tag{19}$$

Again by direct substitution, equation (19) becomes:

$$\begin{aligned} J(N) = \text{minimum}\Bigg\{ & \left[\left\{\phi-\frac{\Delta\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\right\}\left\{\phi\underline{x}(N-2)+\underline{\Delta}u(N-2)\right\}\right]^T Q \\ & \left[\left\{\phi-\frac{\Delta\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\right\}\left\{\phi\underline{x}(N-2)+\underline{\Delta}u(N-2)\right\}\right] \\ & + \left[\phi\underline{x}(N-2)+\underline{\Delta}u(N-2)\right]^T Q \left[\phi\underline{x}(N-2)+\underline{\Delta}u(N-2)\right]\Bigg\} + J(N-2) \end{aligned} \tag{20}$$

Minimizing with respect to u(N-2), remembering that J(N-2) is not a function of u(N-2); and $\partial J(N-2)/\partial u(N-2) = 0$:

$$\frac{\partial J(N)}{\partial u(N-2)} = 2\underline{\Delta}^T Q\phi \underline{x}(N-2) + 2\underline{\Delta}^T Q\underline{\Delta} u(N-2)$$

$$+2\underline{\Delta}^T\{\phi - \frac{\underline{\Delta}\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\}^T \phi^T \underline{x}^T(N-2) Q\{\phi - \frac{\underline{\Delta}\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\}\underline{\Delta}$$

$$+2\{\phi - \frac{\underline{\Delta}\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\}\underline{\Delta}^T Q\{\phi - \frac{\underline{\Delta}\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\}\underline{\Delta}\ u(N-2) = 0 \qquad (21)$$

Let $\psi_1 = \left[I - \frac{\underline{\Delta}\underline{\Delta}^T (Q+P_o)}{\underline{\Delta}^T (Q+P_o)\underline{\Delta}}\right]\phi$, where $P_o = 0$ (22)

therefore

$$\frac{\partial J(N)}{\partial u(N-2)} = \underline{\Delta}^T Q\phi \underline{x}(N-2) + \underline{\Delta}^T \psi_1^T Q\ \psi_1 \underline{\Delta}\underline{x}(N-2)$$

$$+\underline{\Delta}^T Q\underline{\Delta} u(N-2) + \underline{\Delta}^T \psi_1^T Q\psi_1 \underline{\Delta} u(N-2) = 0 \qquad (23)$$

or

$$\therefore\ u(N-2) = -\frac{\underline{\Delta}^T (Q+\psi_1^T Q\psi_1)\phi}{\underline{\Delta}^T (Q+\psi_1^T Q\psi_1)\underline{\Delta}}\ \underline{x}(N-2) \qquad (24)$$

Let $P_1 = \psi_1^T Q\psi_1$ (25)

$$u(N-2) = -\frac{\underline{\Delta}^T (Q+P_1)\phi}{\underline{\Delta}^T (Q+P_1)\underline{\Delta}}\ \underline{x}(N-2) \qquad (26)$$

The optimum control policy for the (N-2) stage has been found represented by equation (26).

To further the establishment of recursion formulas, the next stage

back must be considered and minimized with respect to u(N-3). The following symbology will be utilized in this stage of the development:

$$A \stackrel{\Delta}{=} \left[\phi \underline{x}(N-3)+\underline{\Delta}u(N-3)\right] \tag{27}$$

and

$$B \stackrel{\Delta}{=} \left[\psi_1 \phi A+\psi_1 \underline{\Delta}\left(-\frac{\underline{\Delta}^T(Q+P_1)\phi}{\underline{\Delta}^T(Q+P_1)\underline{\Delta}}\right)A\right] \tag{28}$$

The cost function then becomes:

$$J(N) = \text{minimum } \{A^TQA+B^TQB\}+J(N-3) \tag{29}$$

Let

$$\psi_2 = \left[I-\frac{\underline{\Delta}\underline{\Delta}^T(Q+P_1)}{\underline{\Delta}^T(Q+P_1)\underline{\Delta}}\right]\phi \tag{30}$$

Therefore

$$J(N)= \text{minimum}\{A^TQA+\left[\psi_1\psi_2A\right]^TQ\left[\psi_1\psi_2A\right]\}+J(N-3) \tag{31}$$

Minimizing the cost function with respect to u(N-3):

$$\frac{\partial J(N)}{\partial u(N-3)} = 2\underline{\Delta}^TQ\phi\underline{x}(N-3)+2\underline{\Delta}^TQ\underline{\Delta}u(N-3)$$

$$+2\underline{\Delta}^T\psi_2^T\psi_1^TQ\psi_1\psi_2\phi\underline{x}(N-3)$$

$$+2\underline{\Delta}^T\psi_2^T\psi_1^TQ\psi_1\psi_2\underline{\Delta}u(N-3) = 0 \tag{32}$$

Solving for u(N-3):

$$u(N-3)= -\frac{\underline{\Delta}^TQ\phi+\underline{\Delta}^T\psi_2^T\psi_1^TQ\psi_1\psi_2\phi}{\underline{\Delta}^TQ\underline{\Delta}+\underline{\Delta}^T\psi_2^T\psi_1^T\phi\psi_1\psi_2\underline{\Delta}}\underline{x}(N-3) \tag{33}$$

giving

$$u(N-3) = -\frac{\underline{\Delta}^T(Q+\psi_2^T\psi_1^TQ\psi_1\psi_2)\phi}{\underline{\Delta}^T(Q+\psi_2^T\psi_1^TQ\psi_1\psi_2)\underline{\Delta}}\,\underline{x}(N-3) \tag{34}$$

Let

$$P_2 = \psi_2^T\psi_1^TQ\psi_1\psi_2 \tag{35}$$

Therefore

$$u(N-3) = -\frac{\underline{\Delta}^T(Q+P_2)\,\phi}{\underline{\Delta}^T(Q+P_2)\,\underline{\Delta}}\,\underline{x}(N-3) \tag{36}$$

It is believed that an iterative relationship may now be shown without regressing to another stage. The following equations may readily be shown to be true by an inductive proof.

$$\psi_{k+1} = \left[I - \frac{\underline{\Delta}\underline{\Delta}^T(Q+P_k)}{\underline{\Delta}^T(Q+P_k)\,\underline{\Delta}}\right]\phi \tag{37}$$

where

$$P_k = \psi_k^T\psi^T{}_{k-1}\cdots\psi_2^T\psi_1^TQ\psi_1\psi_2\cdots\psi_{k-1}\psi_k \tag{38}$$

or

$$P_k = \psi_k^TP_{k-1}\psi_k,\quad P_o = 0 \tag{39}$$

and

$$u(N-k) = -\frac{\underline{\Delta}^T(Q+P_{k-1})\,\phi}{\underline{\Delta}^T(Q+P_{k-1})\,\underline{\Delta}}\,\underline{x}\,(N-k) \tag{40}$$

$$\underline{x}(k+1) = \phi\underline{x}(k)+\underline{\Delta}u(k) \tag{41}$$

To illustrate the use of the above equations, the system presented in Chapter I will be considered with the sampling period equal to one second:

$$\phi = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad \underline{\Delta} = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

To find u(k), equations 37, 39, and 40 must be solved:

$$\psi_1 = \phi - \frac{\underline{\Delta}\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}} = \begin{bmatrix} 0 & 0 \\ -2 & -1 \end{bmatrix} \tag{42}$$

and

$$P_1 = \psi_1^T Q\psi_1 = 0 \tag{43}$$

and

$$\therefore u(k) = -\frac{\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}}\underline{x}(k) = -2x_1(k) - 2x_2(k) \tag{44}$$

By substitution into equation (44) and solving for u(0), we find $x_1(1)=0$ and $x_2(1)= -2$. The following phase plane (Figure II-1) results:

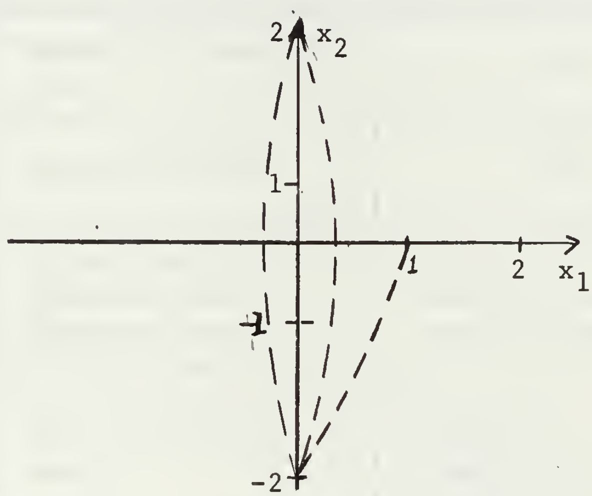

Figure II-1: Phase Plane For Example Problem

This clearly indicates that a limit cycle results and that the cost function, as determined with the use of this weighting matrix, yields a relative minimum cost of one unit.

Using the derived recursion formulas 37, 39, 40 and 41, PROGRAM OPCON1 was written. Figures II-2 through II-10 represent abbreviated flow charts of the program and the subroutines used in the program. It

must be remembered that the arithmetic operations represented in the flow charts are all matrix operations. As can be seen from Figure II-2, the required inputs to the program must be read from data cards. The following table (Table II-1) indicates the order that the data cards are read, the format required on these cards, and the definition of the variable concerned. The primary output of OPCON1 is the value of each state variable, the control signal, and the cost of each stage. A complete program listing may be found in the appendix.

Variable	Definition	Format of Data Card
M	Number of stages plus one	I5
N	Order of system	I5
$\underline{\Delta}$	Column vector of system impulse responses due to control inputs	8F10.6 (read from top of column to bottom)
ϕ	State transition matrix	8F10.6 (read in order by columns)
Q	Weighting matrix of the cost function	8F10.6 (read in order by columns)
$\underline{x}(0)$	Initial conditions on the state vector	8F10.6 (read from top of column to bottom)

Table II-1: Definition and Format of Data Cards for Program OPCON1.

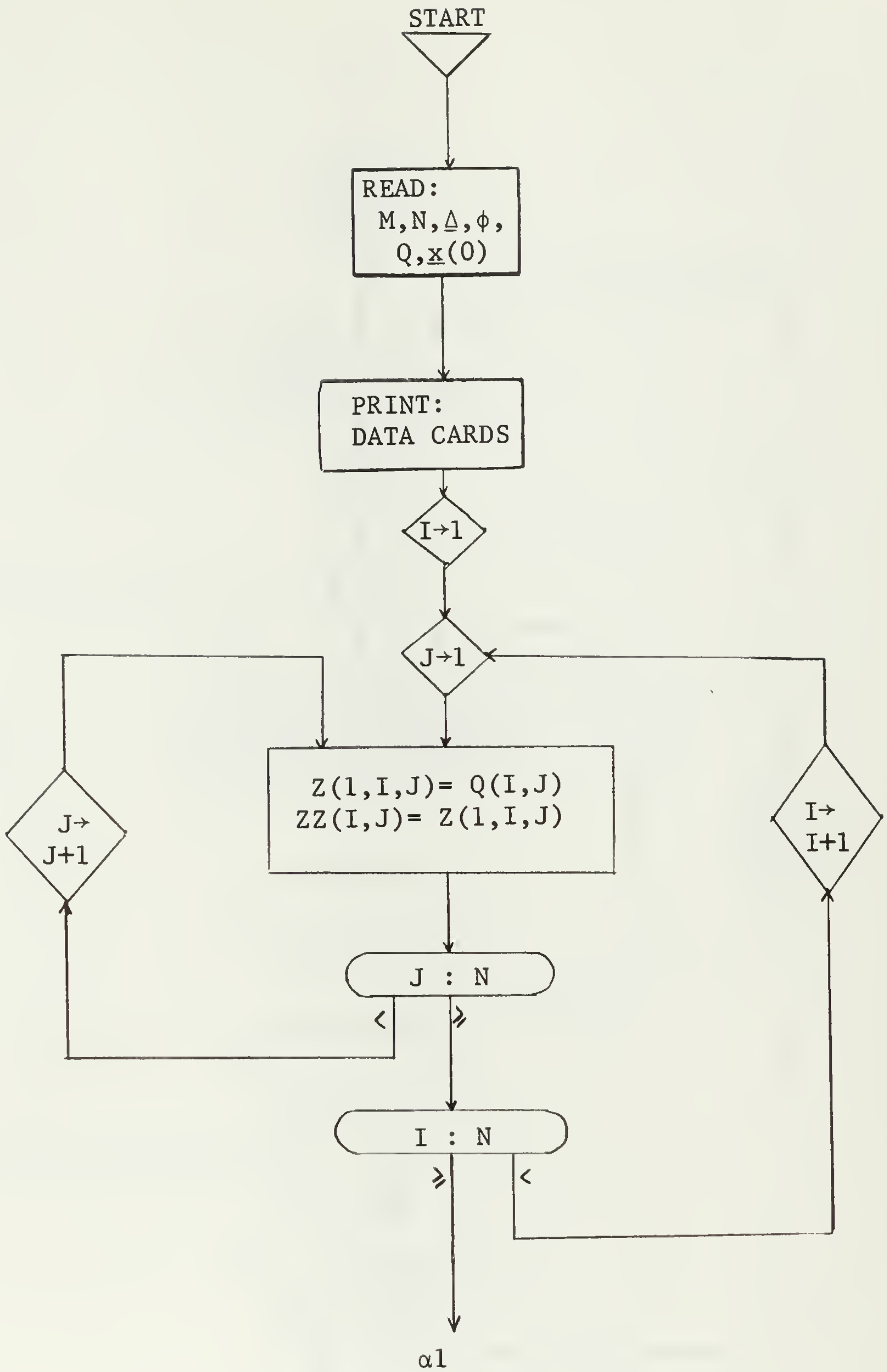

Figure II-2: Flow Chart Of Program OPCON1

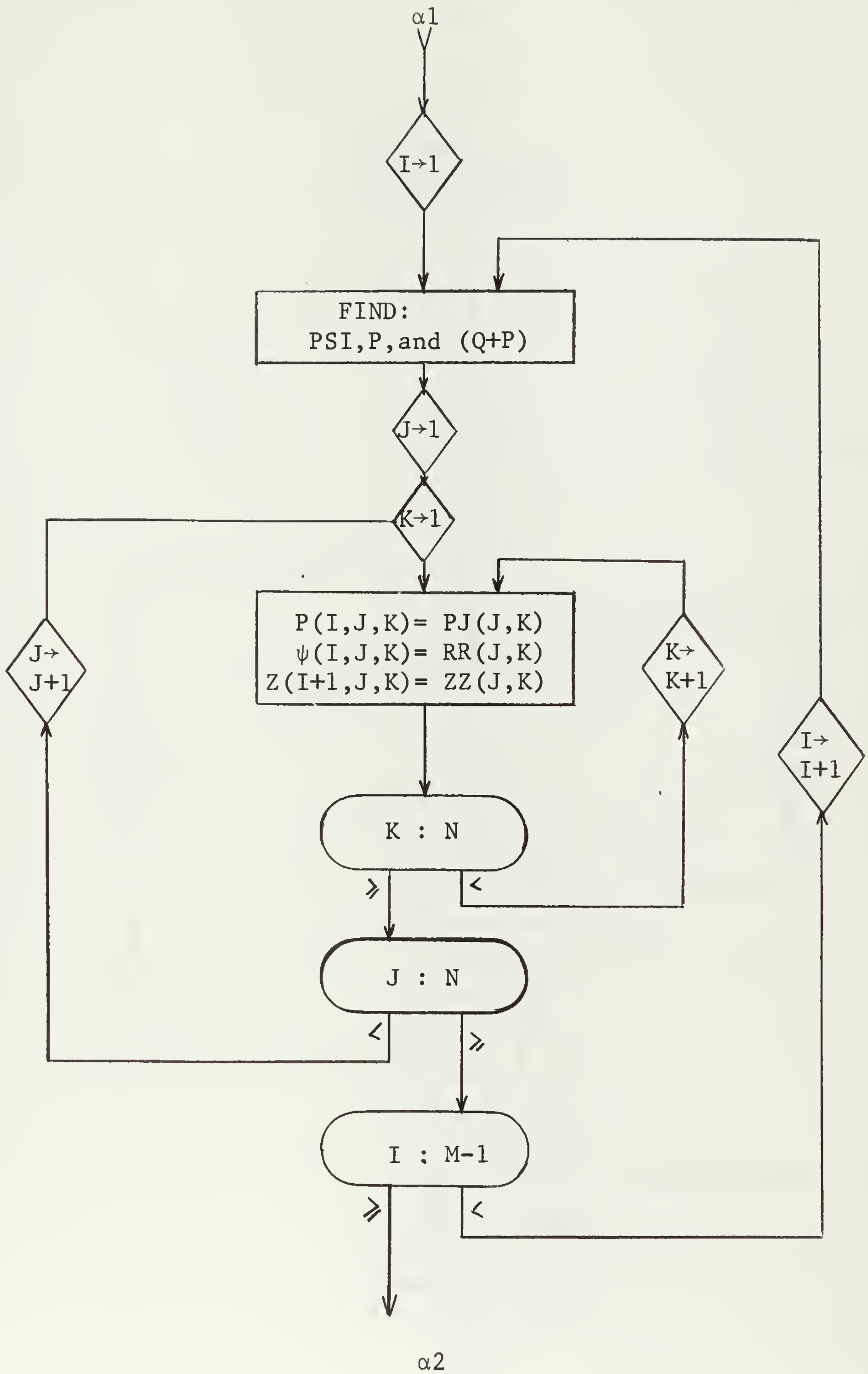

FIGURE II-2: (continued)

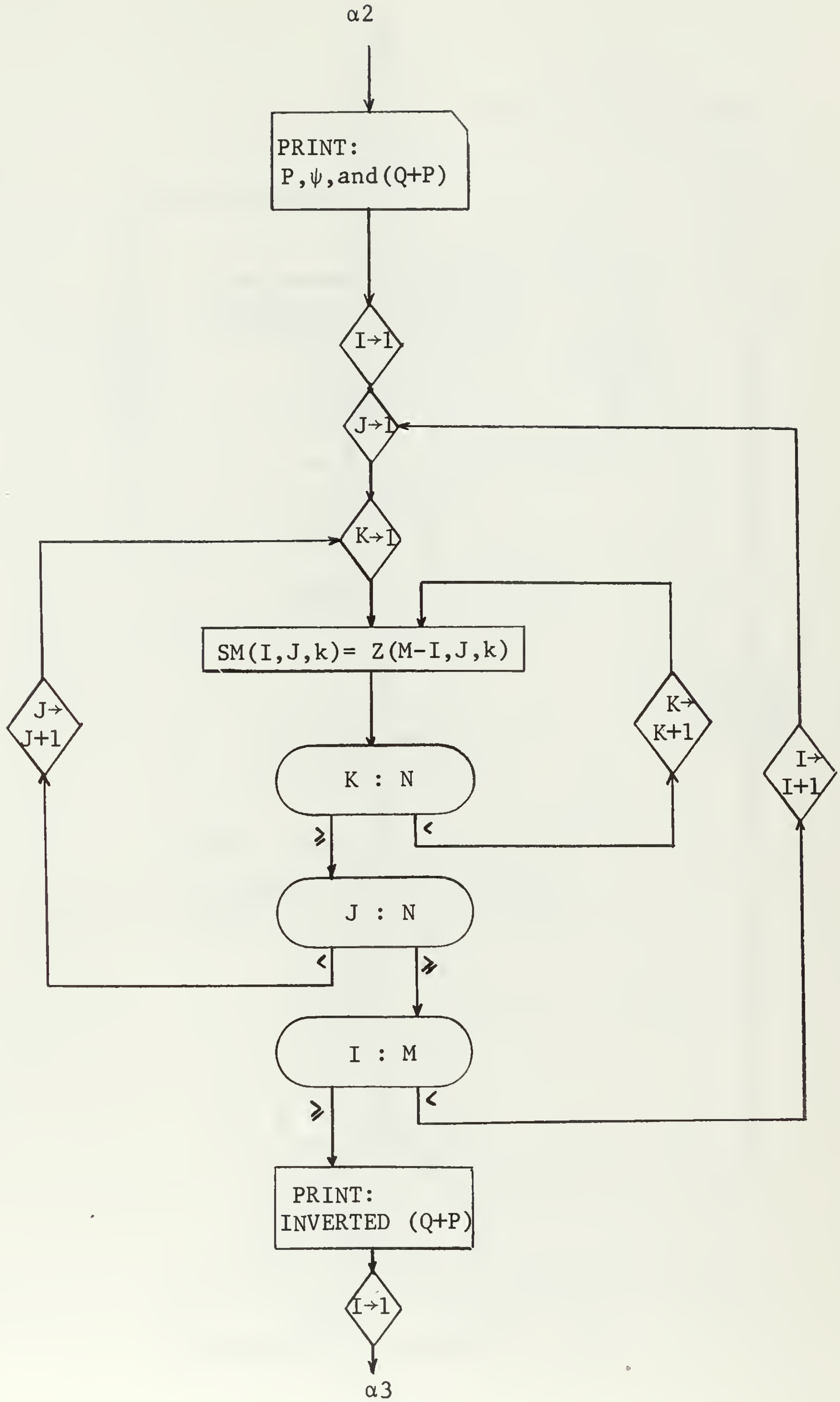

Figure II-2: (continued)

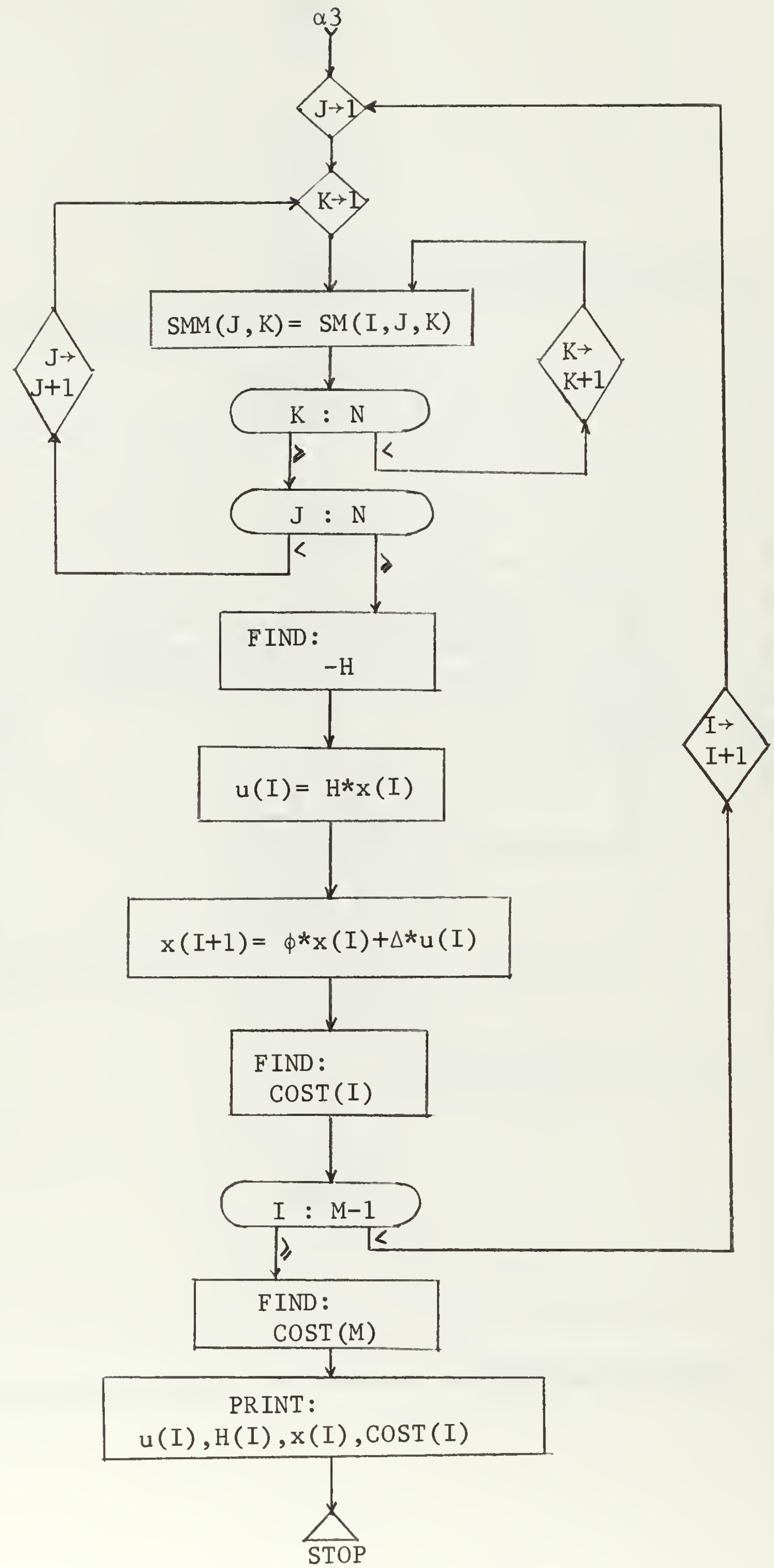

Figure II-2: (concluded)

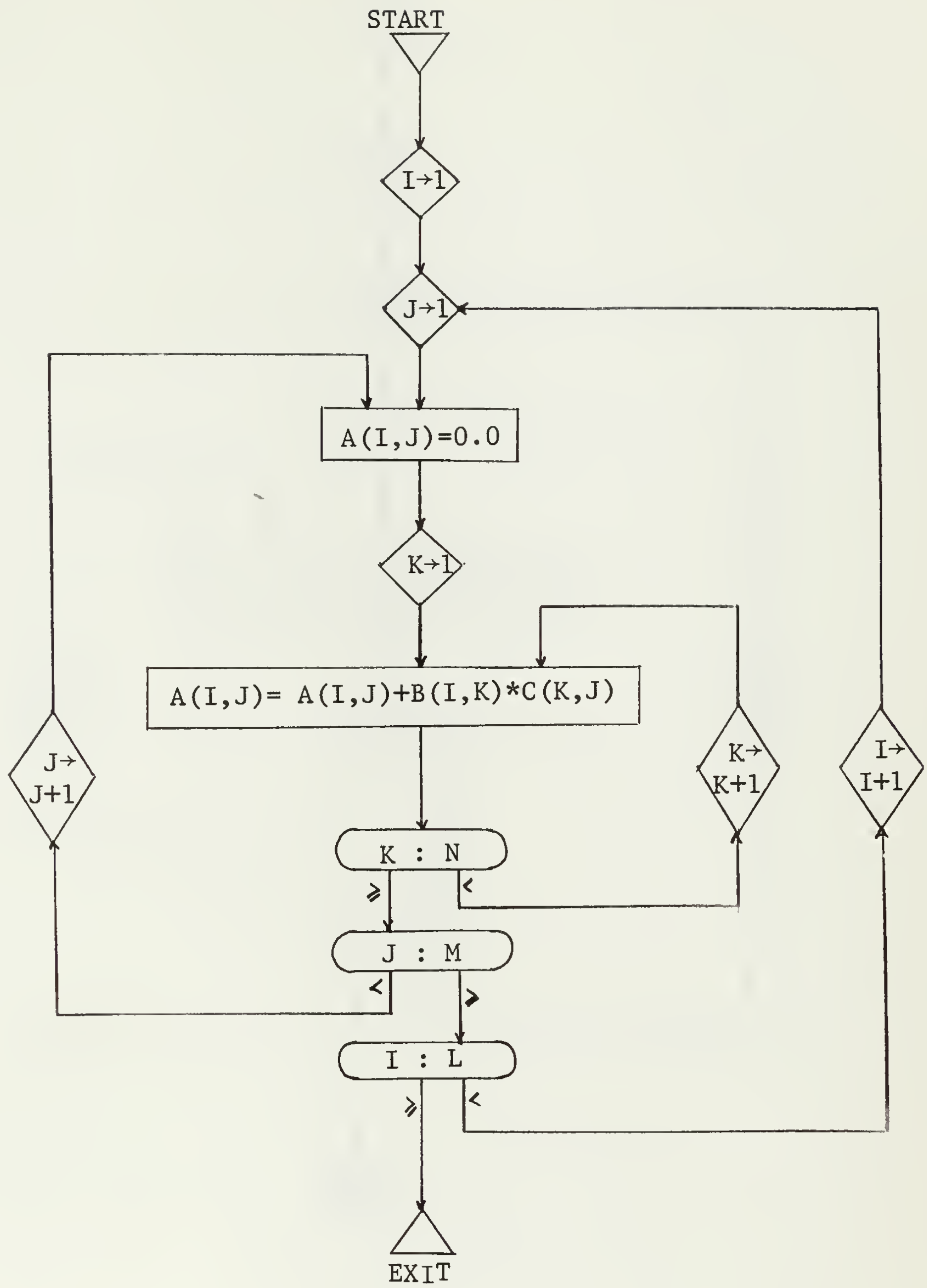

Figure II-3: Flow Chart For Subroutine PROD(A,B,C,L,M,N)

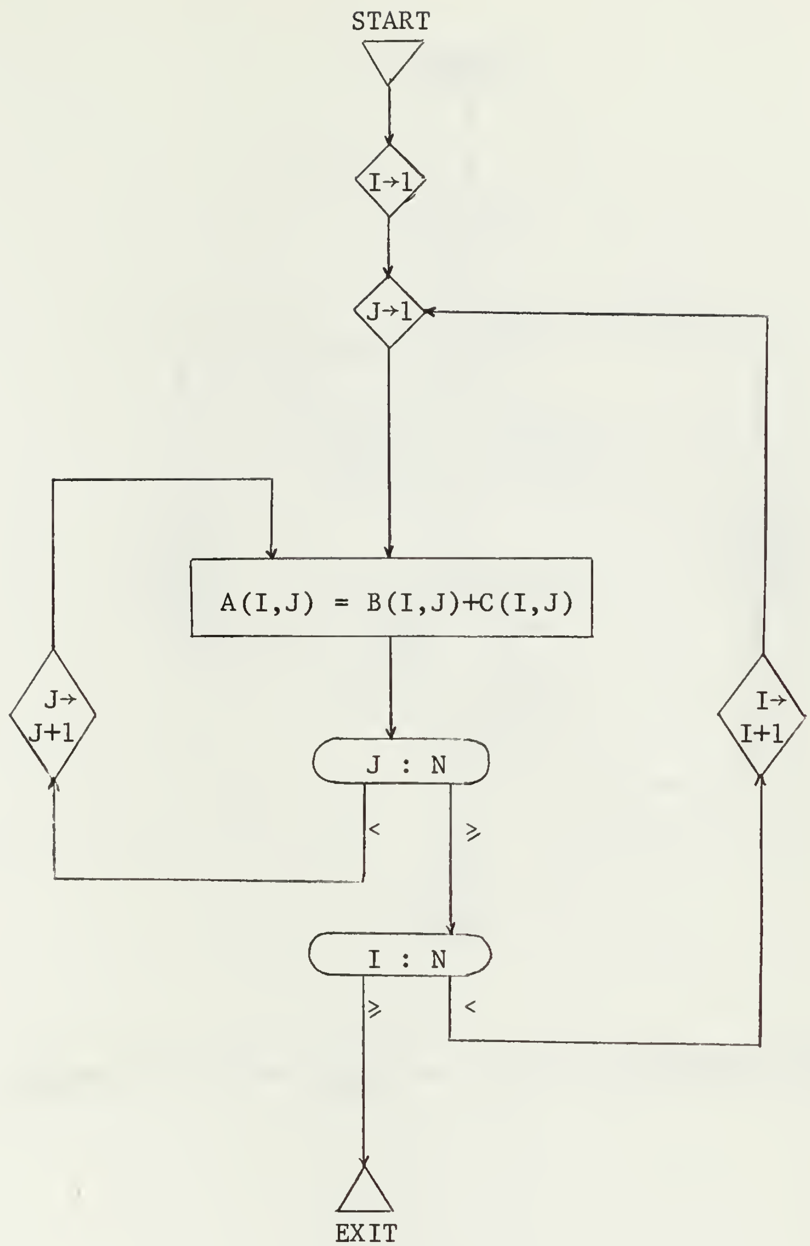

Figure II-4: Flow Chart For Subroutine SUM(A,B,C,N)

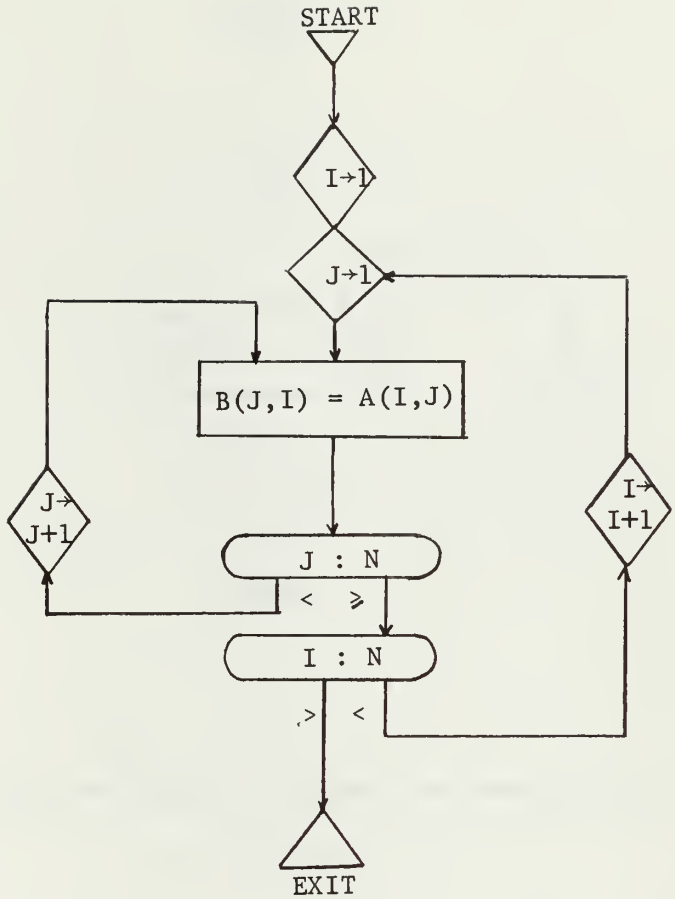

Figure II-5: Flow Chart For Subroutine TRANSQ(A,B,N)

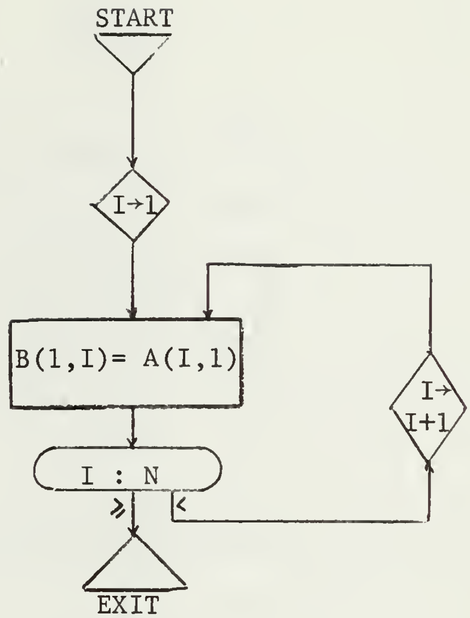

Figure II-6: Flow Chart For Subroutine TRANCOL(A,B,N)

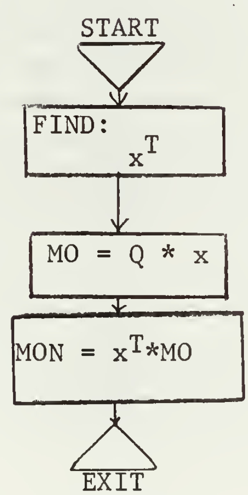

Figure II-7: Flow Chart For Subroutine Money(MON,E,Q,N)

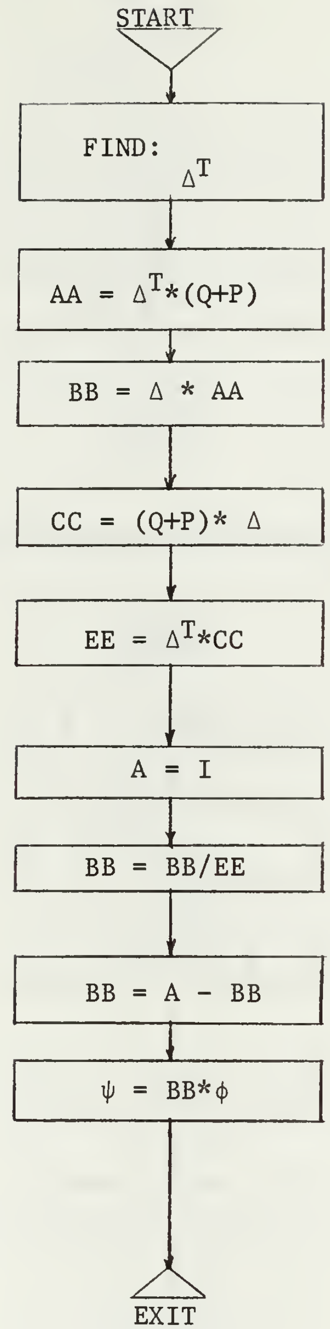

Figure II-8: Flow Chart For Subroutine PPSI(B,C,D,E,F,N)

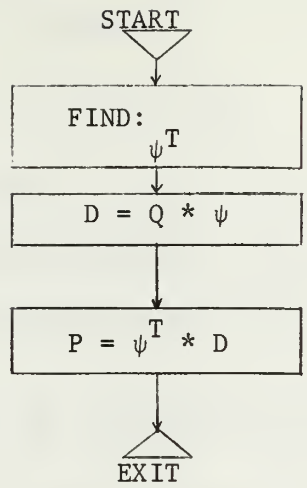

Figure II-9 : Flow Chart For Subroutine PP(A,B,C,N)

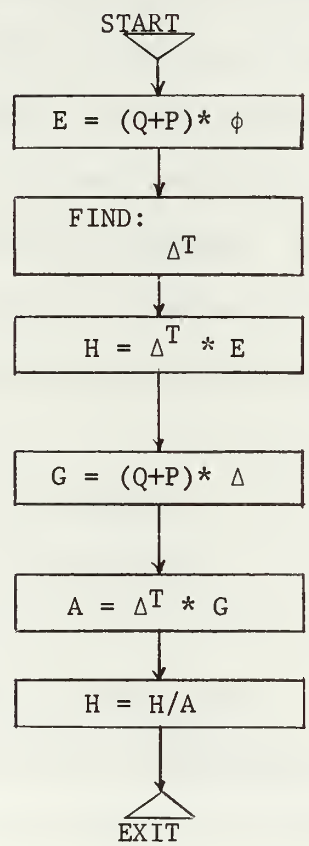

Figure II-10: Flow Chart For Subroutine FF(A,B,C,D,N)

CHAPTER III

PROGRAM OPCON2

Program OPCON2 was written as the final phase for a versatile simulator to determine a constant gain matrix for optimization of a control system according to the following cost function:

$$J(N)=\text{minimum} \sum_{k=1}^{N} (\underline{x}^T(k)Q\underline{x}(k)+ru^2(k-1)) \tag{45}$$

Again u(k) is a scalar control and the Q matrix is determined in the same manner as outlined in the previous chapter. The variable, r, is a weighting constant,(Lagrange multiplier) determined by user needs and acts as a quadratic integral constraint on the control effort; such as a limit on energy necessary to provide the desired control. By letting r equal zero, we have the cost function of OPCON1 and identical results are obtained from the two programs. With the use of OPCON1, no restraints are put on the control required by the system which may result in the necessity of an unlimited supply of fuel to obtain the desired optimal results.

Again let us consider the system:

$$\dot{\underline{x}} = F\underline{x} + \underline{D}u \tag{1'}$$

whose discrete solution is:

$$\underline{x}(k+1)= \phi\underline{x}(k)+\underline{\Delta}u(k) \tag{4'}$$

where u(k) and $\underline{\Delta}$ are defined in the same manner as in the previous chapter.

By starting with the last stage first, equation 45 becomes:

$$J(N)=\text{minimum}\left\{\underline{x}^T(N)Q\underline{x}(N)+ru^2(N-1)\right\} +J(N-1) \tag{46}$$

or by substitution of equation 4':

$$J(N)=\text{minimum}\left\{(\phi\underline{x}(N-1)+\underline{\Delta}u(N-1))^T Q(\phi\underline{x}(N-1)+\underline{\Delta}u(N-1))+ru^2(N-1)\right\} +J(N-1) \tag{47}$$

We now minimize J(N) with respect to u(N-1) noting that J(N-1) is not a function of u(N-1).

$$\frac{\partial\ J(N)}{\partial u(N-1)} = 0 = \underline{\Delta}^T Q\phi\underline{x}(N-1)+\underline{\Delta}^T Q\underline{\Delta}u(N-1)+ru(N-1) \qquad (48)$$

or,

$$u(N-1) = -\ \frac{\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}+r}\ \underline{x}(N-1) \qquad (49)$$

As in Chapter II, the denominator of equation 49 will always be a scalar number; therefore, no matrix inversion is required.

By proceeding back one more stage, the control policy u(N-2) may be optimized. Equation 45 becomes:

$$J(N)=\text{minimum}\left[\underline{x}^T(N)Q\underline{x}(N)+\underline{x}^T(N-1)Q\underline{x}(N-1)+ru^2(N-1)\ +ru^2(N-2)\right]+J(N-2) \qquad (50)$$

The following symbology will be used for clarity in this minimization:

$$A \triangleq \phi\underline{x}(N-2)+\underline{\Delta}u(N-2) \qquad (51)$$

and,

$$B \triangleq -\frac{\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}+r} \qquad (52)$$

By direct substitution using 4', 51, and 52, equation 50 becomes:

$$J(N)=\text{minimum}\left\{(\phi A+\underline{\Delta}BA)^T Q(\phi A+\underline{\Delta}BA)+A^T QA+r(BA)^T(BA)+ru^2(N-2)\right\}+J(N-2) \qquad (53)$$

By rearranging terms,

$$J(N)=\text{minimum}\left\{A^T QA+((\phi+\underline{\Delta}B)A)^T Q((\phi+\underline{\Delta}B)A)+r(BA)^2+ru^2(N-2)\right\}+J(N-2) \qquad (54)$$

We now let,

$$a_1^T = B = -\ \frac{\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}+r} \qquad (55)$$

and

$$\psi_1 = \phi+\underline{\Delta}a_1^T \qquad (56)$$

Equation 54 now becomes:

$$J(N)=\text{minimum}\left\{A^T QA+A^T\psi_1^T Q\psi_1 A+r(a_1^T A)^2+ru^2(N-2)\right\}+J(N-2) \qquad (57)$$

We now minimize J(N) with respect to u(N-2),

$$\frac{\partial\ J(N)}{\partial u(N-2)} = 0 = \underline{\Delta}^T QA+\underline{\Delta}^T \psi_1^T Q\psi_1 A+r\, a_1 A a_1^T \underline{\Delta}+ru(N-2) \tag{58}$$

Solving for u(N-2) by substitution of 51 into equation 58,

$$u(N-2) = -\frac{\left\{\underline{\Delta}^T Q+\underline{\Delta}^T \psi_1^T Q\psi_1+r\underline{\Delta}^T a_1 a_1^T\right\}\phi}{\left\{\underline{\Delta}^T Q\underline{\Delta}+\underline{\Delta}^T \psi_1^T Q\psi_1 \underline{\Delta}+r\underline{\Delta}^T a_1 a_1^T \underline{\Delta}+r\right\}}\ \underline{x}(N-2) \tag{59}$$

We now let,

$$P_1 = \psi_1^T Q\psi_1+Q+ra_1 a_1^T \tag{60}$$

Equation 59 becomes:

$$u(N-2) = -\frac{\underline{\Delta}^T P_1 \phi}{\underline{\Delta}^T P_1 \underline{\Delta}+r}\ x(N-2) \tag{61}$$

As we did in the previous iteration, we let,

$$a_2^T = -\frac{\underline{\Delta}^T P_1 \phi}{\underline{\Delta}^T P_1 \underline{\Delta}+r} \tag{62}$$

and,

$$\psi_2 = \phi+\underline{\Delta} a_2^T \tag{63}$$

By proceeding back one more stage, a recursive relationship may be defined. We shall define the following quantity for brevity in this development:

$$G \triangleq \phi\underline{x}(N-3)+\underline{\Delta}u(N-3) \tag{64}$$

The cost function, 45, may be written as,

$$J(N) = \text{minimum}\Big[\underline{x}^T(N)Q\underline{x}(N)+\underline{x}^T(N-1)Q\underline{x}(N-1)+\underline{x}^T(N-2)Q\underline{x}(N-2)$$

$$+ru^2(N-1)+ru^2(N-2)+ru^2(N-3)\Big]+J(N-3) \tag{65}$$

By substitution of 4', 55, 56, 62, 63, and 64, equation 65 becomes:

$$J(N)=\text{minimum}\Big\{(\psi_2\psi_1 G)^T Q(\psi_2\psi_1 G)+r(\psi_2 a_1^T G)^T(\psi_2 a_1^T G)$$

$$+(\psi_2 G)^T G(\psi_2 G)+r(a_2^T G)^T(a_2^T G)$$

$$+G^T QG+ru^2(N-3)\Big\}+J(N-3) \tag{66}$$

Minimizing the cost over N stages with respect to u(N-3),

$$\frac{\partial J(N)}{\partial u(N-3)}=0=\underline{\Delta}^T\psi_2^T\psi_1^T Q\psi_1\psi_2 G+\underline{\Delta}^T\psi_2^T Q\psi_2 G+\underline{\Delta}^T QG$$

$$+r\underline{\Delta}^T\psi_2^T a_1 a_1^T\psi_2 G+r\underline{\Delta}^T a_2 a_2^T G+ru(N-3) \tag{67}$$

Substituting 64 into equation 67 and solving for u(N-3), we find the optimum control policy:

u(N-3) =

$$-\frac{\left\{\underline{\Delta}^T\psi_2^T\psi_1^T Q\psi_1\psi_2+\underline{\Delta}^T\psi_2^T Q\psi_2+\underline{\Delta}^T Q+r\underline{\Delta}^T\psi_2^T a_1 a_1^T\psi_2+r\underline{\Delta}^T a_2 a_2^T\right\}\phi}{\left\{\underline{\Delta}^T\psi_2^T\psi_1^T Q\psi_1\psi_2\underline{\Delta}+\underline{\Delta}^T\psi_2^T Q\psi_2\underline{\Delta}+\underline{\Delta}^T Q\underline{\Delta}+r\underline{\Delta}^T\psi_2^T a_1 a_1^T\psi_2\underline{\Delta}+r\underline{\Delta}^T a_2 a_2^T\underline{\Delta}+r\right\}}\ \underline{x}(N-3) \tag{68}$$

Combining terms,

$$u(N-3)=-\frac{\left\{\underline{\Delta}^T\psi_2^T(\psi_1^T Q\psi_1+Q+ra_1 a_1^T)\psi_2+\underline{\Delta}^T Q+r\underline{\Delta}^T a_2 a_2^T\right\}\phi}{\left\{\underline{\Delta}^T\psi_2^T(\psi_1^T Q\psi_1+Q+ra_1 a_1^T)\psi_2\underline{\Delta}+\underline{\Delta}^T Q\underline{\Delta}+r\underline{\Delta}^T a_2 a_2^T\underline{\Delta}+r\right\}}\ \underline{x}(N-3) \tag{69}$$

From equation 60, equation 69 becomes:

$$u(N-3)=-\frac{\left\{\underline{\Delta}^T\psi_2^T P_1\psi_2+\underline{\Delta}^T Q+r\underline{\Delta}^T a_2 a_2^T\right\}\phi}{\left\{\underline{\Delta}^T\psi_2^T P_1\psi_2\underline{\Delta}+\underline{\Delta}^T Q\underline{\Delta}+r\underline{\Delta}^T a_2 a_2^T\underline{\Delta}+r\right\}}\ \underline{x}(N-3) \tag{70}$$

We now let,

$$P_2=\psi_2^T P_1\psi_2+Q+ra_2 a_2^T \tag{71}$$

and equation 70 becomes:

$$u(N-3)=-\frac{\underline{\Delta}^T P_2\phi}{\underline{\Delta}^T P_2\underline{\Delta}+r}\ \underline{x}(N-3) \tag{72}$$

Let,

$$a_3^T = -\frac{\underline{\Delta}^T P_2 \phi}{\underline{\Delta}^T P_2 \underline{\Delta}+r} \tag{73}$$

and,

$$\psi_3 = \phi+\underline{\Delta}a_3^T \tag{74}$$

As in Chapter II, a recursive relationship may now be shown without proceeding back another stage and may be shown to be true by an inductive proof. Let

$$a_k^T = -\frac{\underline{\Delta}^T P_{k-1}\phi}{\underline{\Delta}^T P_{k-1}\underline{\Delta}+r}, \qquad a_o^T = 0 \tag{75}$$

and,

$$\psi_k = \phi+\underline{\Delta}a_k^T, \qquad \psi_o = 0 \tag{76}$$

where,

$$P_k = \psi_k^T P_{k-1}\psi_k+Q+ra_k a_k^T, \quad P_o=Q \tag{77}$$

and,

$$u(N-k) = a_k^T \underline{x}(N-k) \tag{78}$$

from,

$$\underline{x}(k+1) = \phi\underline{x}(k)+\underline{\Delta}u(k) \tag{79}$$

By letting r equal zero and solving the same illustrative example as in the previous chapter, the same results are obtained as illustrated below.

$$a_1^T = \frac{-\underline{\Delta}^T Q\phi}{\underline{\Delta}^T Q\underline{\Delta}+r} = \begin{bmatrix} -2.0 & -2.0 \end{bmatrix} \tag{80}$$

and,

$$\psi_1 = \phi+\underline{\Delta}a_1^T = \begin{bmatrix} 0 & 0 \\ -2.0 & -1.0 \end{bmatrix} \tag{81}$$

and,

$$P_1 = \psi_1^T Q\psi_1+Q+ra_1 a_1^T = Q = \begin{bmatrix} 1.0 & 0 \\ 0 & 0 \end{bmatrix} \tag{82}$$

and therefore,

$$u(k)=a_1^T\underline{x}(k)= - 2x_1(k)-2x_2(k) \tag{83}$$

Equation 83 indicates the same results as in Chapter II and perscribes the same phase plane (Figure II-1).

Through the use of the recursion formulas 75, 76, 77, 78, and 79, PROGRAM OPCON2 was written. Figures III-1 through III-5 and Figures II-3 through II-6 represent abbreviated flow charts of the program and the subroutines used in the program.

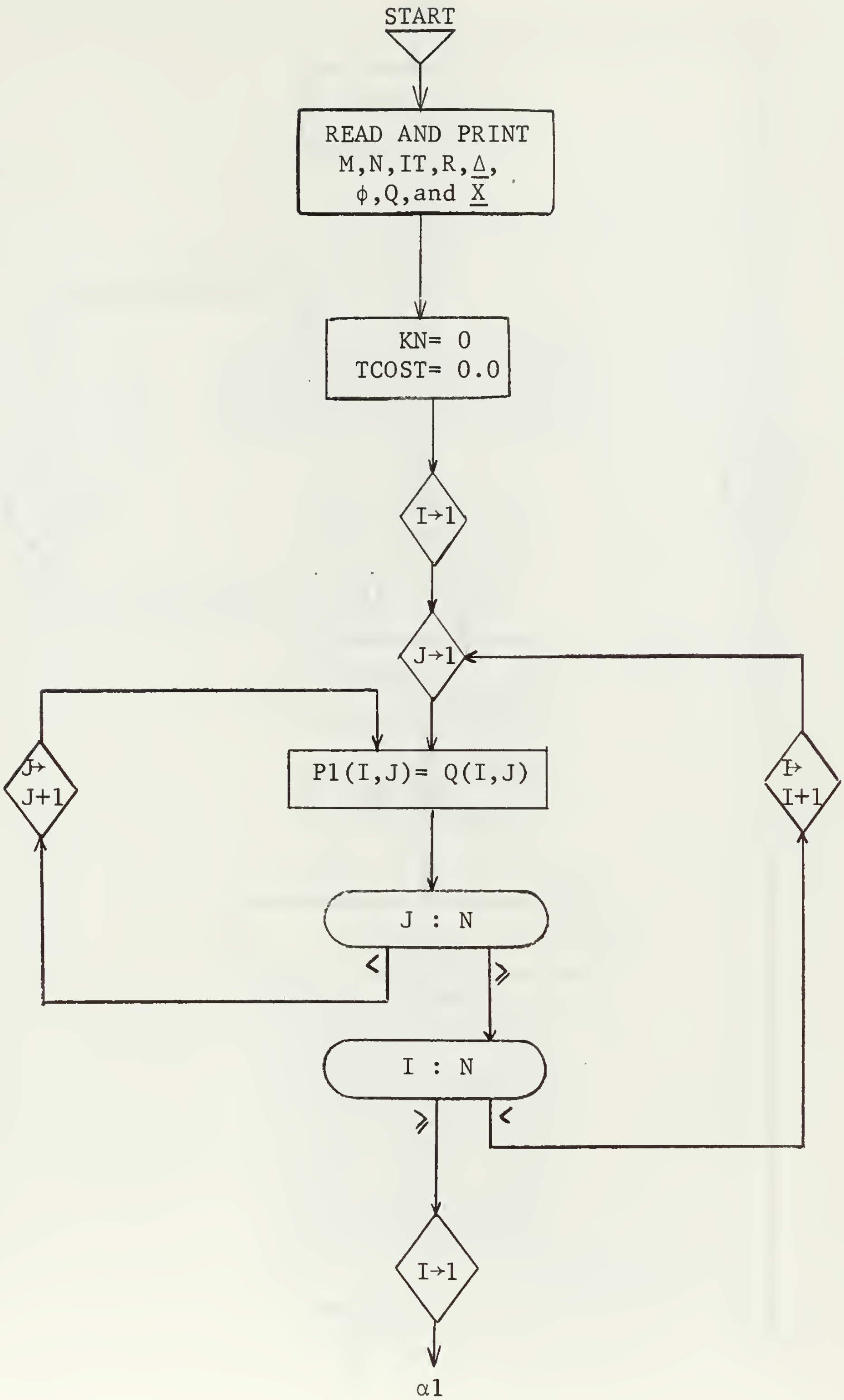

Figure III-1: Flow Chart for PROGRAM OPCON2.

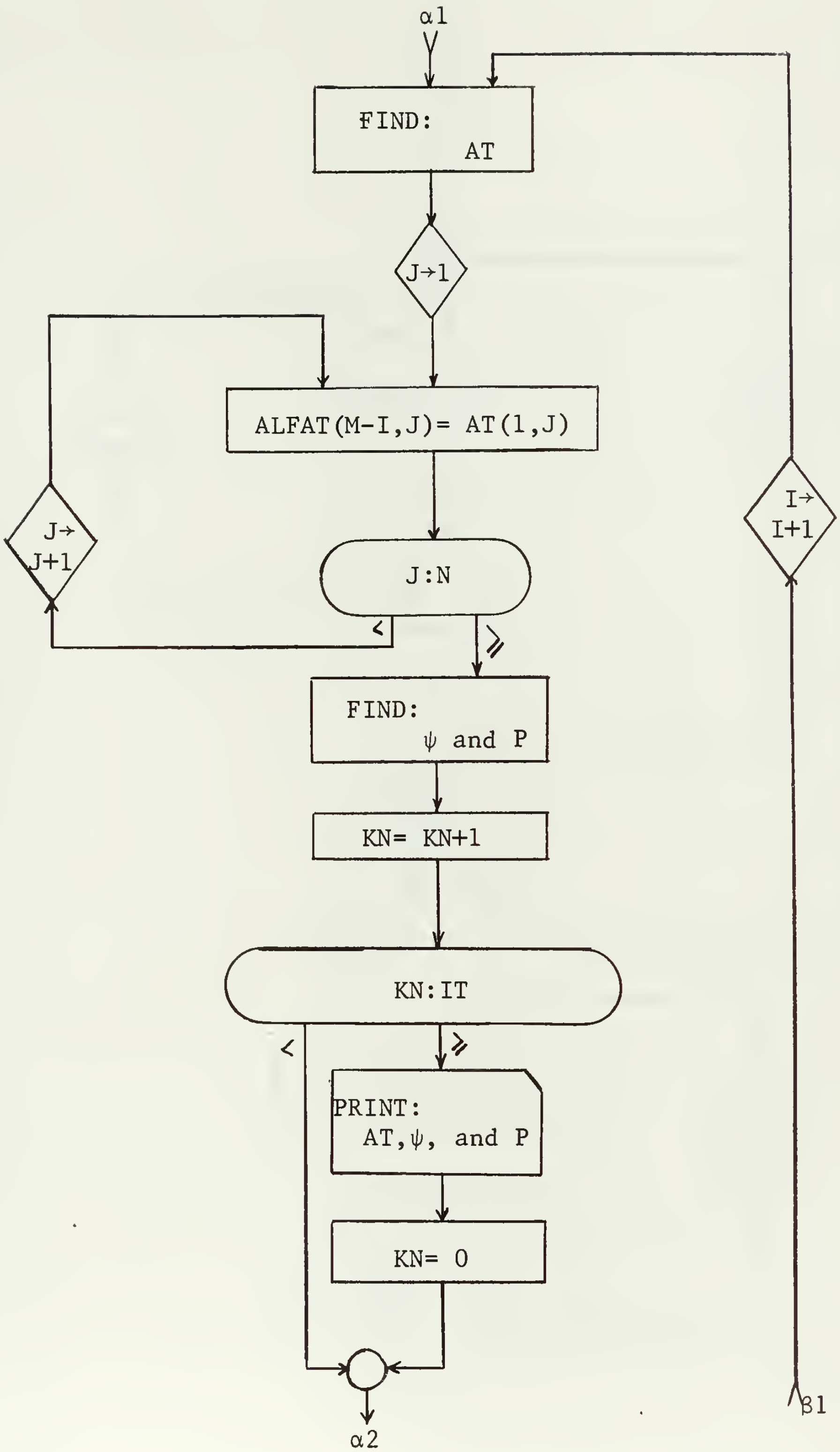

Figure III-1: (continued)

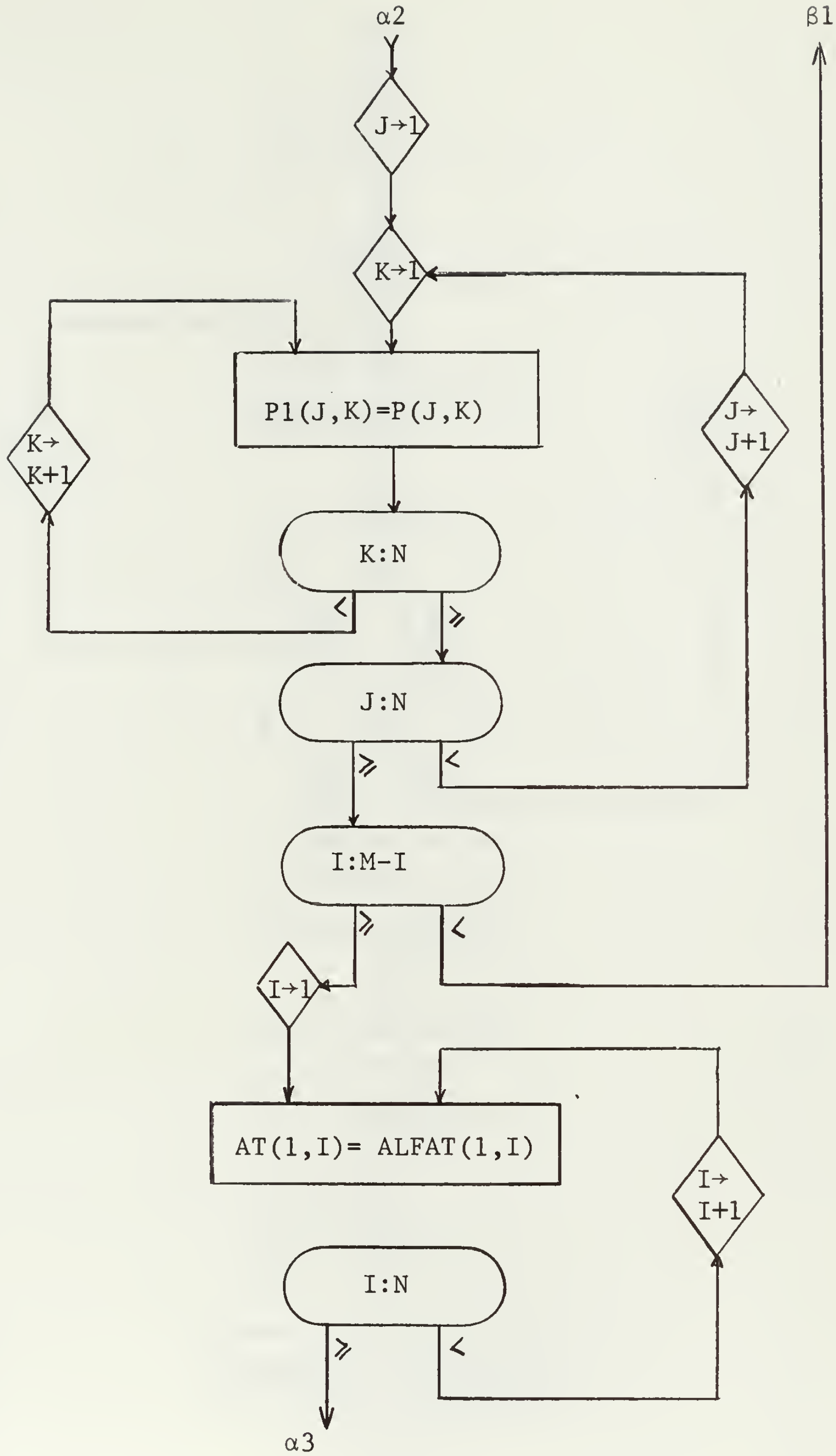

Figure III-1: (continued)

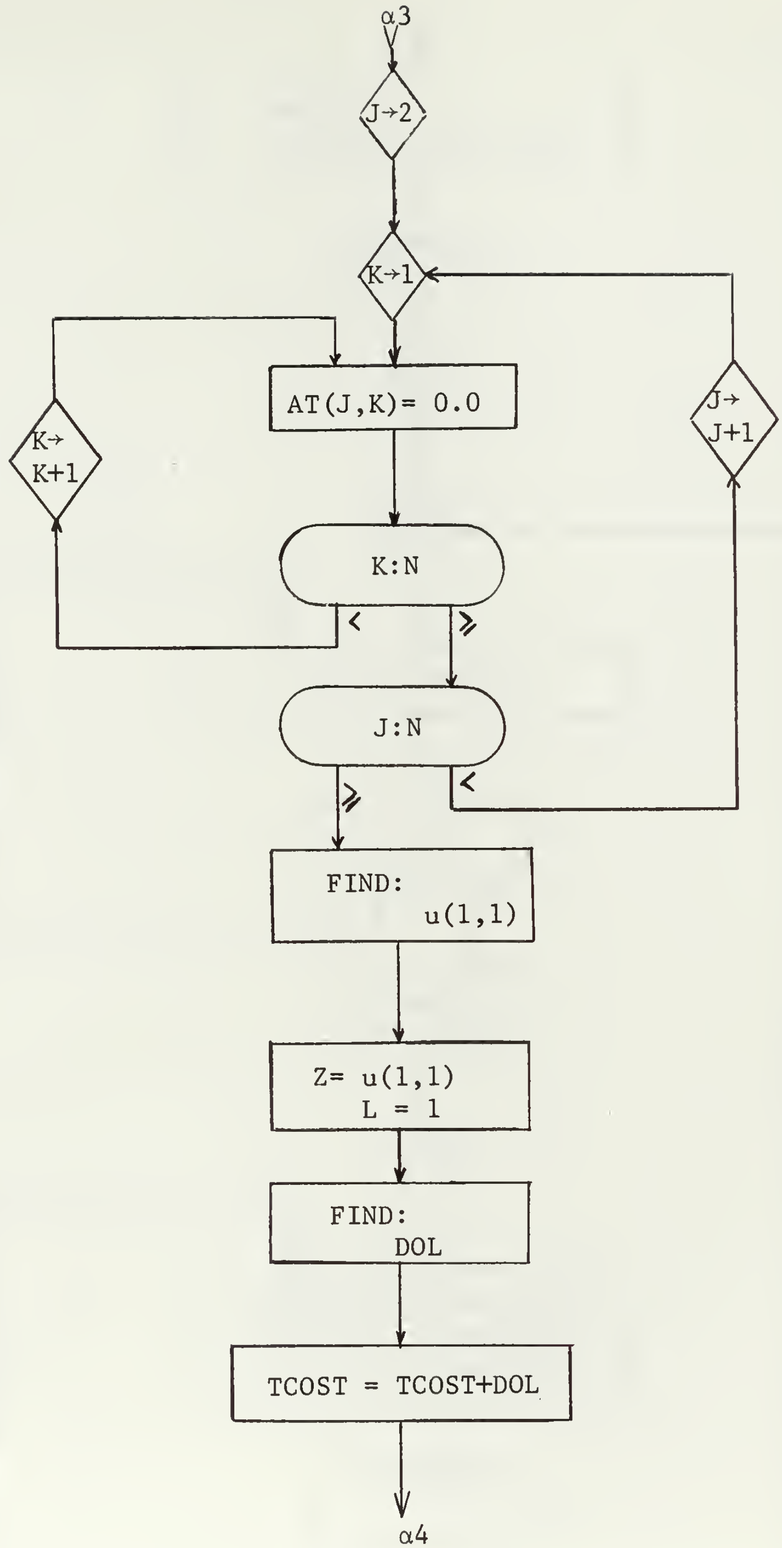

Figure III-1: (continued)

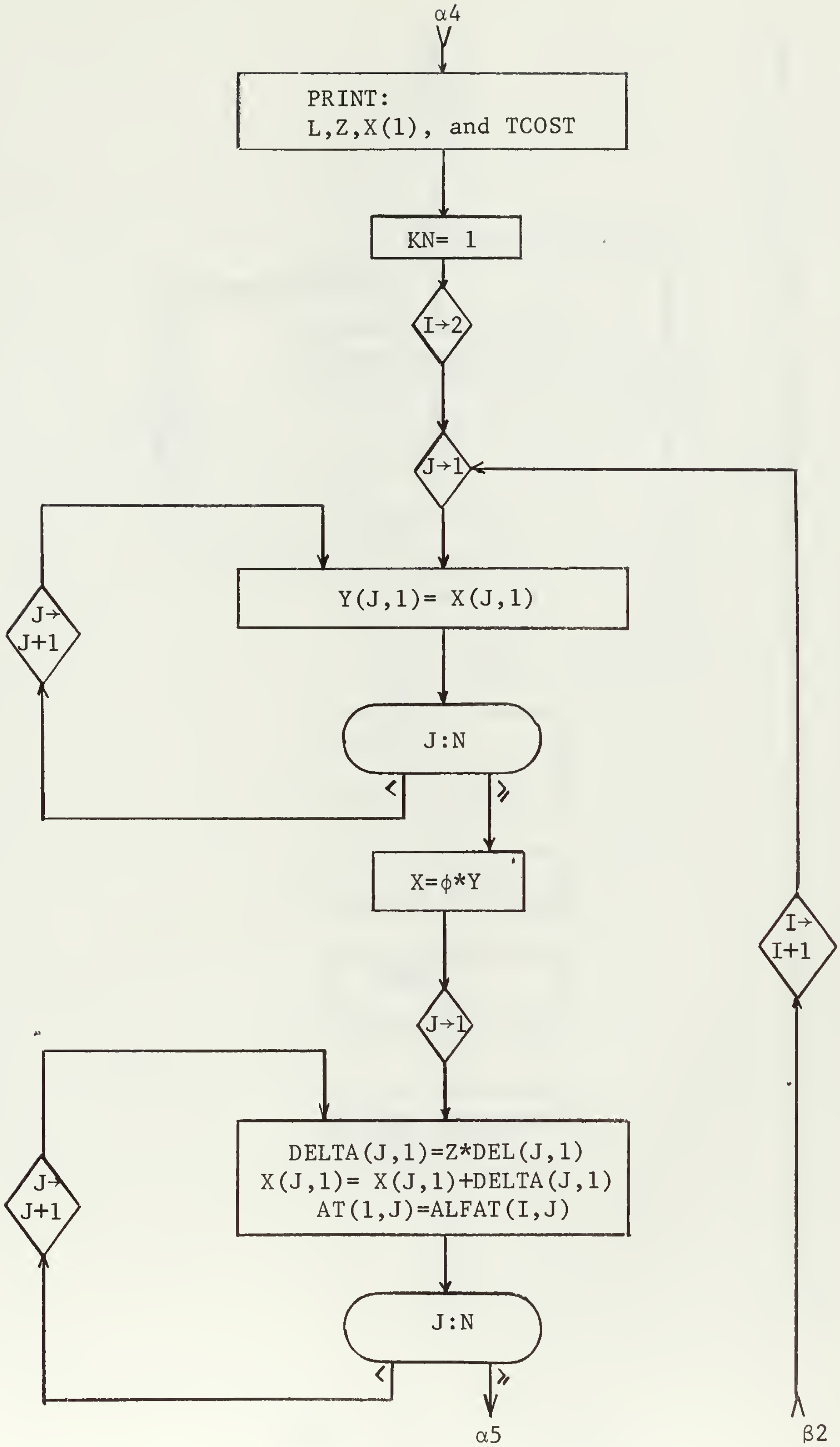

Figure III-1: (continued)

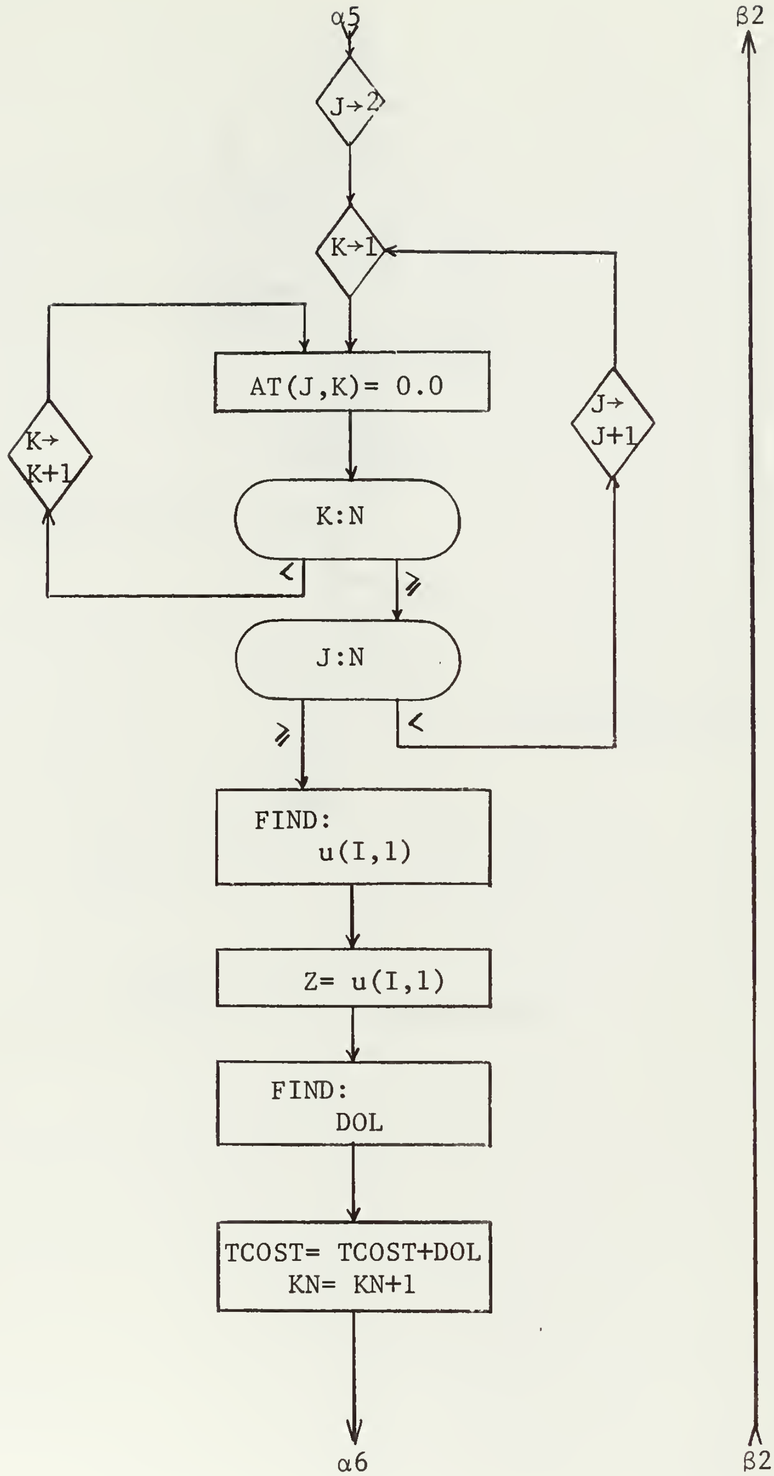

Figure III-1: (continued)

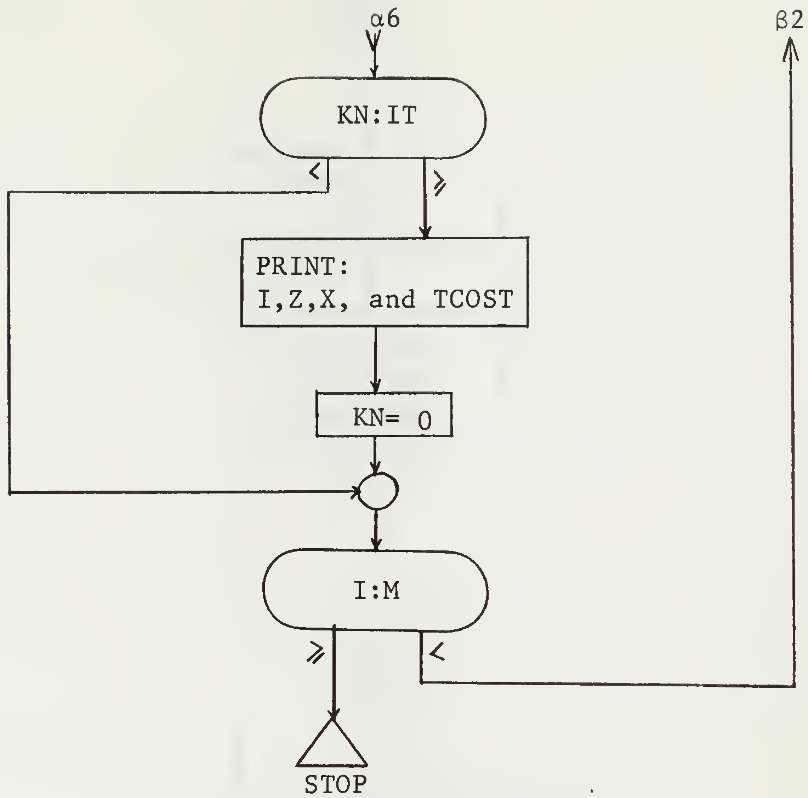

Figure III-1: (continued)

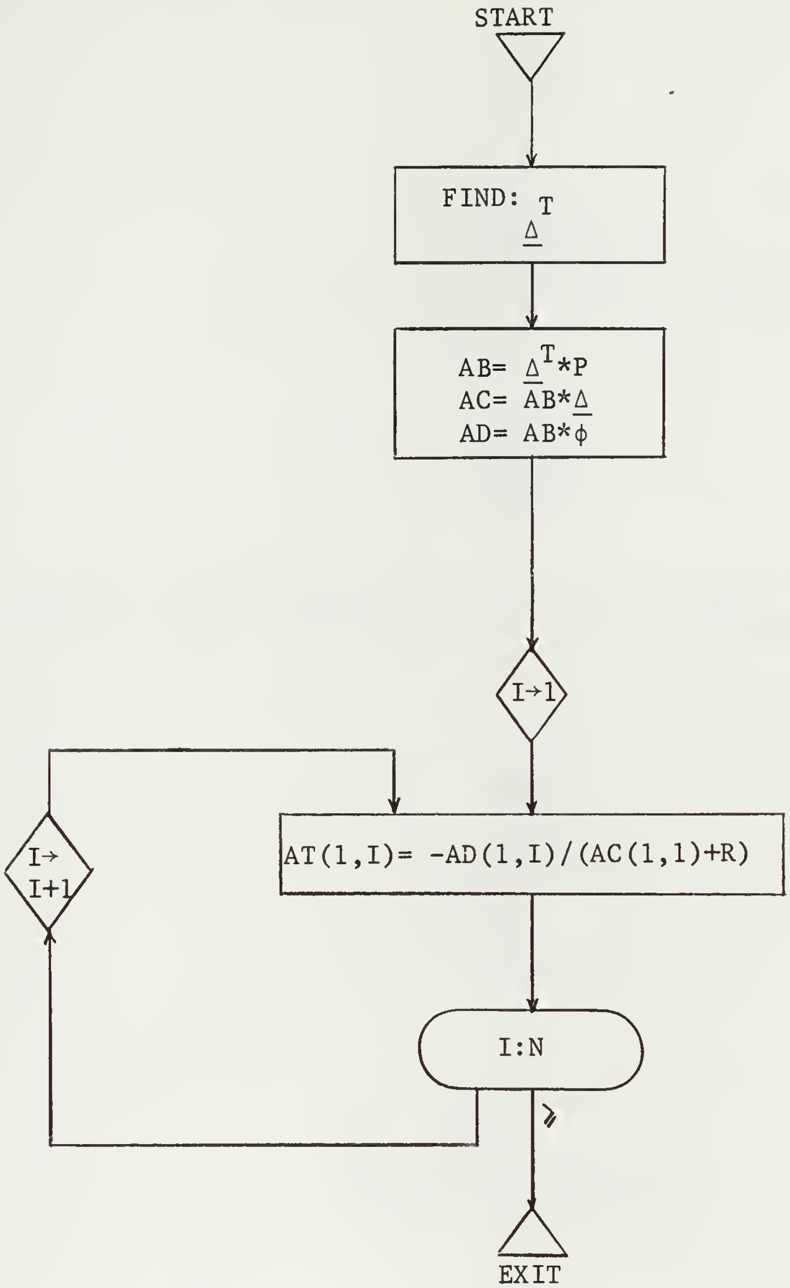

Figure III-2: Flow Chart for SUBROUTINE ATRAN(AT,P,PHI,DEL,R,N)

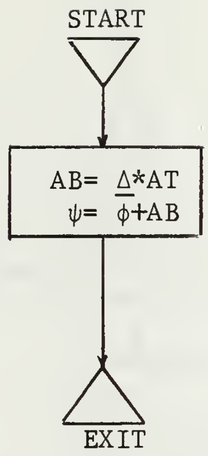

Figure III-3: Flow Chart for SUBROUTINE PPSI(PSI,PHI,DEL,AT,N)

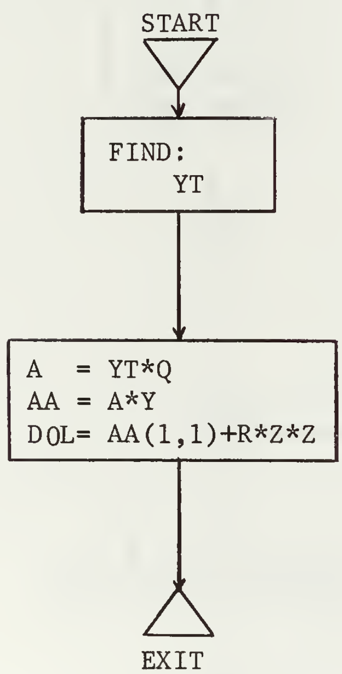

Figure III-4: Flow Chart for SUBROUTINE COST(DOL,Y,Q,R,Z,N)

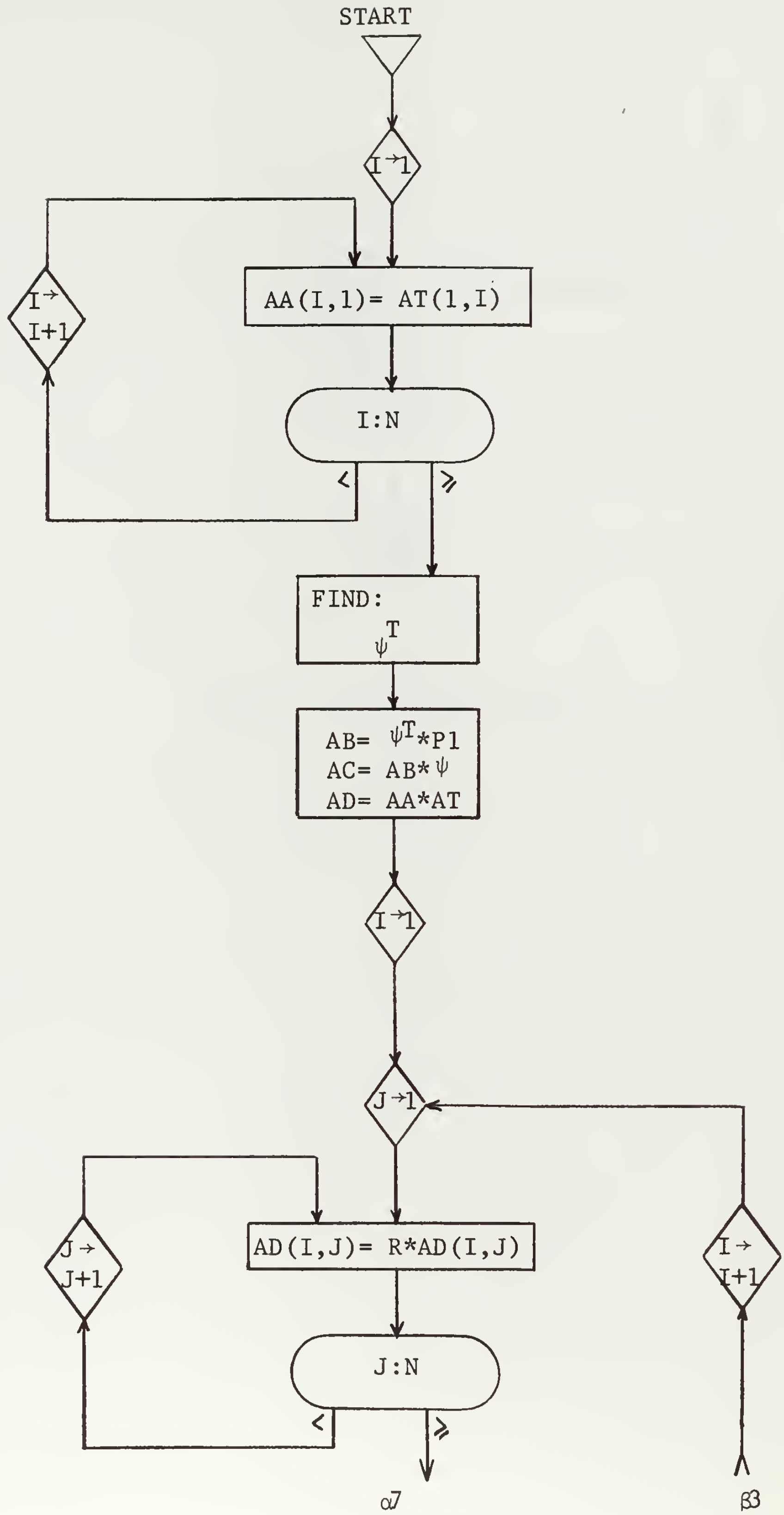

Figure III-5: Flow Chart for SUBROUTINE PP(P,PSI,P1,Q,AT,R,N)

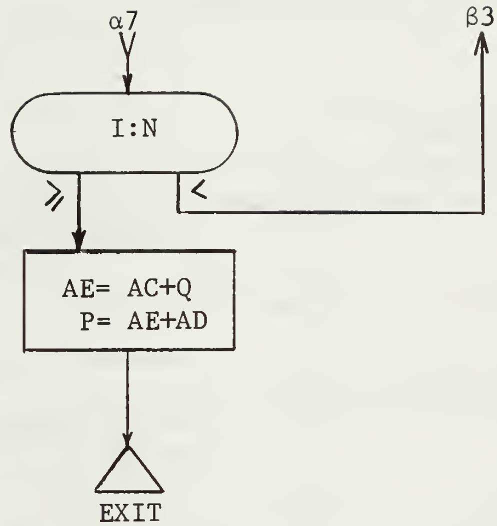

Figure III-5: (concluded)

Figure III-1 shows that the inputs to the program must be read from data cards. Table III-1 shows the order in which these cards are read, the format required on each card, and the definitions of the variables concerned.

Variable	Definition	Format of Data Card
M	Number of stages Plus one	I9
N	Order of system	I9
IT	Frequency of the desired printing of output	I9
r	Weighting constant of cost function	F10.6
$\underline{\Delta}$	Column vector of system impulse responses due to control inputs	8F10.6 (read from top to bottom of column)
ϕ	State transition matrix	nF10.6 (read in a row at a time)
Q	Weighting matrix of the cost function	nF10.6 (read in a row at a time)
$\underline{x}(0)$	Initial conditions on the state vector	8F10.6 (read from top to bottom of column)

Table III-1: Definitions and Format of Data Cards for PROGRAM OPCON2.

The output of PROGRAM OPCON2 is the value of each state variable, the control signal, and the total cost of all previous stages including the present stage. The amount of printed output may be varied by the use of the IT variable's input data card. If all output is to be printed, then IT must be set to one by the data card. If the user wishes to have 1,000 stages and has a requirement of print out of only a sampling of these stages, IT should be set accordingly. By letting IT equal ten, only every tenth stage will be printed out. Another valuable output is the value of a_K^T which approaches the constant gain matrix necessary for optimal control of a continuous system. This is achieved by using a small sample period and many iterations thereby approaching the simulation of a continuous system. OPCON2 is limited for this use because the computer memory size limits the program to approximately 2,000 stages. If the user has no need for the outputs of the values of the control signals, state variables and total costs, then the program is unlimited in its number of stages because the value of a_K^T may be printed out as often as desired without storing it in memory. This alteration has been made in PROGRAM OPCON3, and a listing of this program may be found in the appendix.

The versatility of this program may be further shown in using it to simulate an optimum control in accordance with the cost function,

$$J(N) = \text{minimum}\left[\underline{X}^T(N)\underline{X}(N)\right] \tag{84}$$

which we shall call the final value criterion. Using the same method of derivation as before, we find the following recursion formulas:

$$\psi_K = \phi + \underline{\Delta} a_K^T, \quad \psi_o = 0 \tag{85}$$

and,

$$a_K^T = -\frac{\underline{\Delta}^T P_{K-1} \phi}{\underline{\Delta}^T P_{K-1} \underline{\Delta}}, \quad a_o^T = 0 \tag{86}$$

where,

$$P_{K-1} = \psi_{K-1}^T P_{K-2} \psi_{K-1}, P_o = I \tag{87}$$

and,

$$u(N-K) = a_K^T \underline{X}(N-K) \tag{88}$$

To use OPCON2 for this cost function, we need only to set P_o equal to the identity matrix, Q to the null matrix, and r equal to zero.

As an example of this cost function, we shall again use the same illustrative system. We find that:

$$a_1^T = -\frac{\underline{\Delta}^T \phi}{\underline{\Delta}^T \underline{\Delta}} \quad \begin{bmatrix} -.4 & -1.2 \end{bmatrix} \tag{89}$$

and,

$$\psi_1 = \phi + \underline{\Delta} a_1^T = \begin{bmatrix} .8 & .4 \\ -.4 & -.2 \end{bmatrix} \tag{90}$$

and,

$$P_1 = \psi_1^T \psi_1 = \begin{bmatrix} .8 & .4 \\ .4 & .2 \end{bmatrix} \tag{91}$$

and,

$$a_2^T = -\frac{\underline{\Delta}^T P_1 \phi}{\underline{\Delta}^T P_1 \underline{\Delta}} = \begin{bmatrix} -1.0 & -1.5 \end{bmatrix} \tag{92}$$

therefore,

$$u(0) = -X_1(0) - 1.5X_2(0) \tag{93}$$

Given initial conditions of $X_1(0) = 1.0$ and $X_2(0) = 0.0$, equation 93 results in,

$$u(0) = -1.0 \tag{94}$$

therefore,

$$X_1(1) = X_1(0)+0.5u(0) = 0.5 \tag{95}$$

and,

$$X_2(1) = X_2(0)+u(0) = -1.0 \tag{96}$$

From equation 89,

$$u(1) = -.4X_1(1) - 1.2X_2(1) = 1.0 \tag{97}$$

and,

$$X_1(2) = X_1(1)+X_2(1)+.5u(1) = 0.0 \tag{98}$$

and,

$$X_2(2) = X_2(1)+u(1) = 0.0 \tag{99}$$

Figure III-6 is the phase plane resulting from the calculated controls and shows that the final states at N equal two are definitely at a minimum and the cost function equals zero.

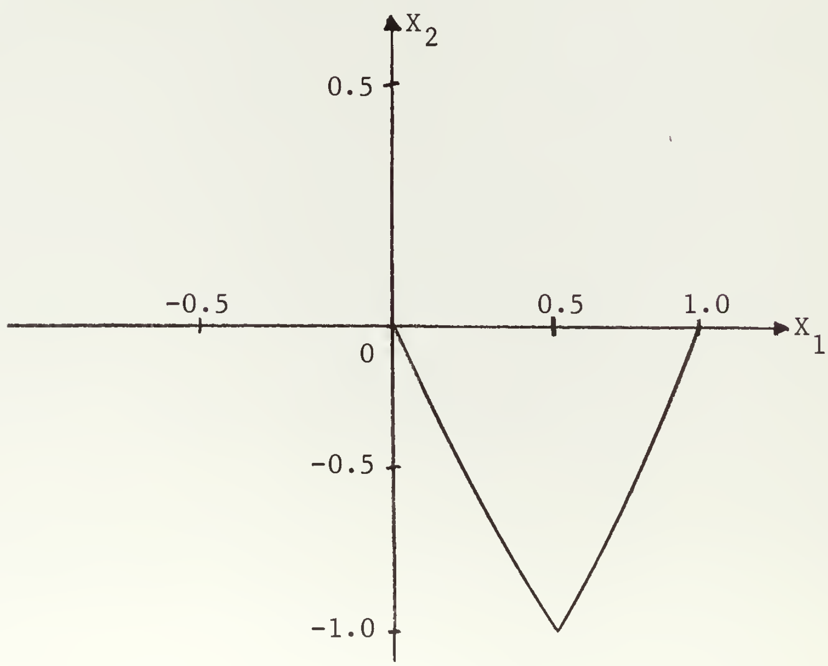

Figure III-6: Phase Plane for Example Problem.

With the use of OPCON2 and OPCON3, many control systems may be simulated. Using these two programs, a system may be designed for a number of optimization criteria while greatly reducing design time. In the next chapter, a typical system will be considered assuming an optimization criterion and the resulting design will be obtained.

CHAPTER IV

EXAMPLE PROBLEM

To show the use of the programs that were written and developed in the preceding chapters, the following example problem was chosen. The transfer function is typical of a servo motor, the roll attitude control of a missile, and many others. Figure IV-1 shows the open loop plant to be controlled.

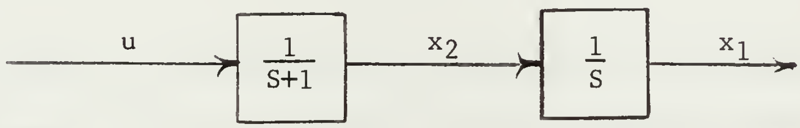

Figure IV-1: Block Diagram of the Open Loop Control System.

The following state equation defines the system:

$$\underline{\dot{x}} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad (100)$$

With the use of PROGRAM PHIDEL and a sample period of one second, ϕ and $\underline{\Delta}$ are:

$$\underline{x}(k+1) = \begin{bmatrix} 1.00 & .632 \\ 0.0 & .368 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} .368 \\ .632 \end{bmatrix} u(k) \qquad (101)$$

To show the flexibility of PROGRAM OPCON2, four different cost functions will be considered and compared. The first cost function will be the terminal value function for time optimal control. (Case I)

$$J(N) = \text{minimum } \underline{x}^T(N)\underline{x}(N) \qquad (84')$$

This will then be compared to the terminal value control with an energy control shown by equation (102). (Case II)

$$J(N) = \text{minimum } \underline{x}^T(N)\underline{x}(N) + \sum_{k=1}^{k=N} u^2(K-1) \qquad (102)$$

The next case of interest minimizes the sum of the squares of the states with an energy limitation. This is represented by equation (45'). (Case III)

$$J(N) = \text{minimum } \sum_{k=1}^{k=N} \underline{x}^T(K)Q\underline{x}(K) + ru^2(K-1) \qquad (45')$$

where,

$$Q = I \quad \text{and} \quad r = 1.0 \qquad (103)$$

The final cost function will also be that of minimizing the states with an energy limitation; however, by use of PROGRAM OPCON3, a steady state feedback matrix was found. These values were then used in PROGRAM OPCON2 to obtain a transient solution. (Case IV) The elements of this control are:

$$a^T(1,1) = -0.627797 \qquad (104)$$

and,

$$a^T(1,2) = -0.615037 \qquad (105)$$

With the use of PROGRAM OPCON2, a transient solution was obtained in each of the four cases. Case II was computed for a twenty second trajectory in the phase plane, Figure IV-3. Cases III and IV were each run for ten second trajectories while Case I was optimized in two samples.

Figure IV-2 is a plot of control versus time for each of the cases. It is of interest to note that both cases III and IV utilize the same controls during the first seventy percent of their trajectories; only when the control is very small in the last couple of sampling points does the control required differ. This thereby leads to identical trajectories in the phase plane indicating that constant gain amplifiers in the

feedback loop are just as effective as variable gain amplifiers for compensation of this plant. Case II requires nearly a constant control effort after the initial control jump has been made. The required control for the twenty second trajector is approximately equal to one half of the necessary control for a ten second trajectory.

Figure IV-3 is a phase plane plot of the four cases. One thing to note here is the trajectory of case II. As the total number of sample periods is increased, the necessary control is decreased as indicated above. This results in the trajectory approaching the x_1 axis as the number of sample points increases indefinitely. This means that the cost function used is optimized when the final value of the states is zero and the control is nearly zero. The point in the phase plane would drift in requiring only a very slight amount of control to stop it at its steady state value.

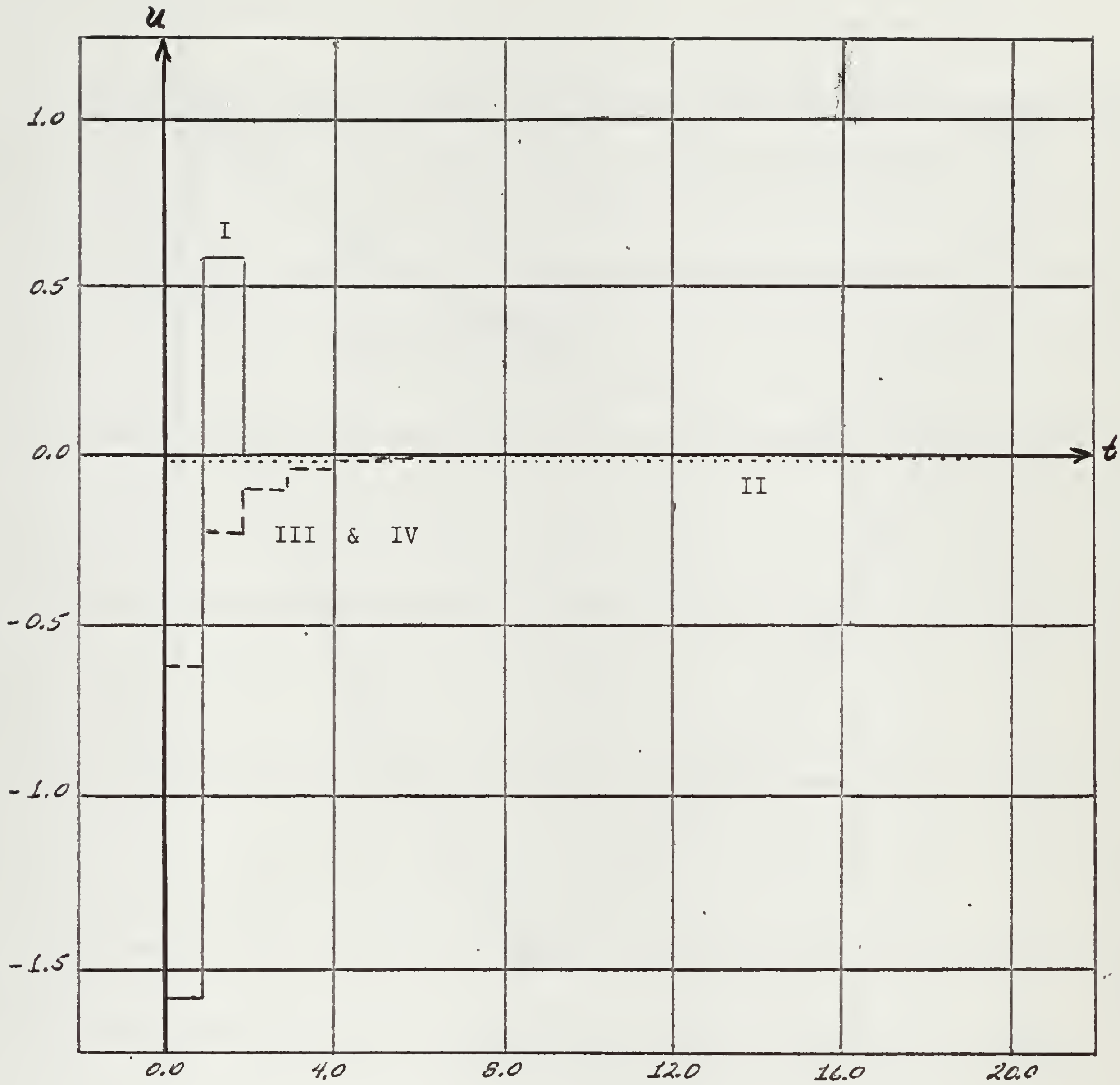

Figure IV-2: Plot of Control Versus Time for the Four Cases.

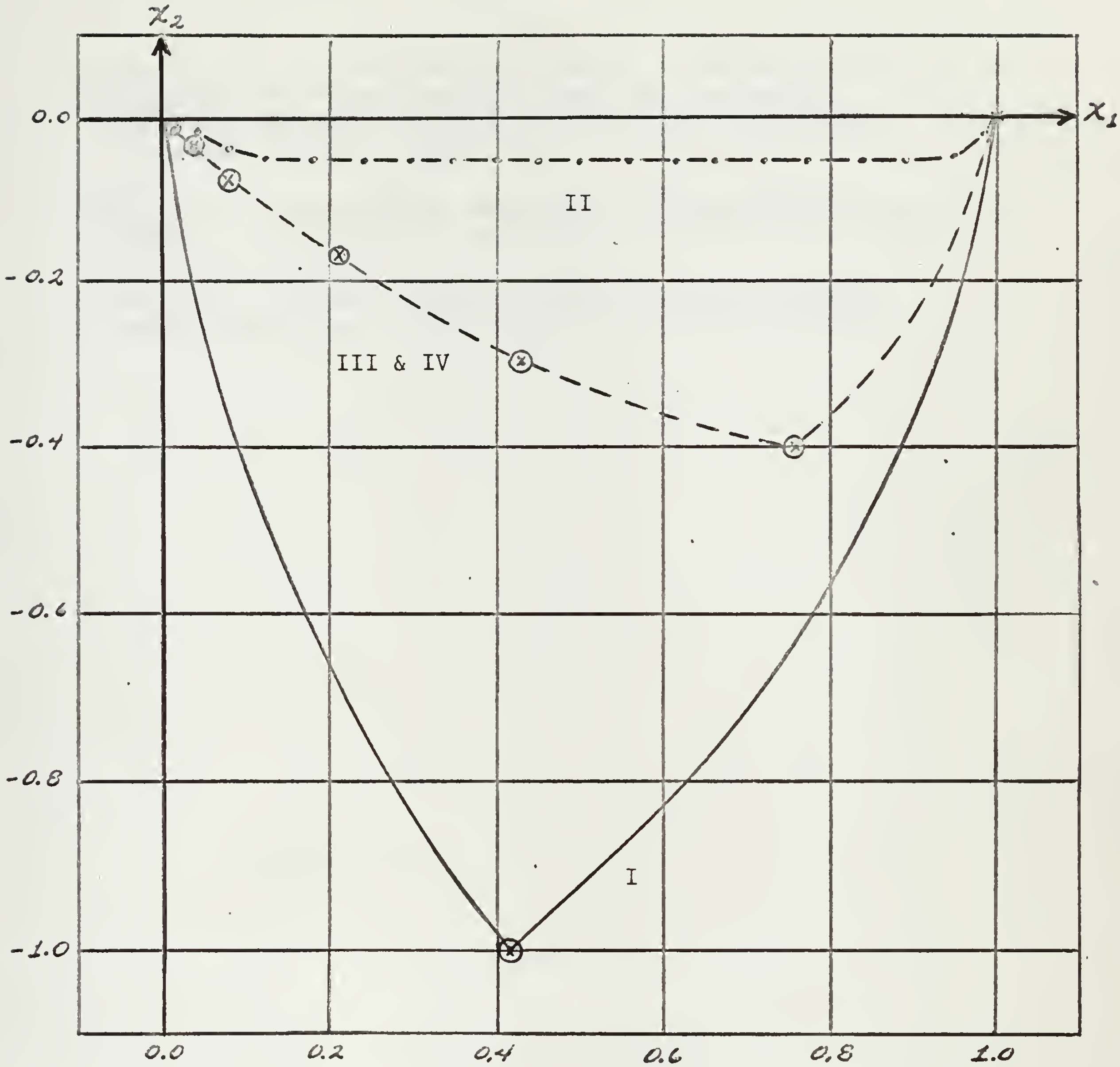

Figure IV-3: Phase Plane Plot of the Four Cases.

BIBLIOGRAPHY

1. Bertram, J. E.: The Concept of State in the Analysis of Discrete-Time Control Systems. American Institute of Electrical Engineers Workshop on State Space Techniques for Control Systems, 1962.

2. Kuo, B. C.: "Analysis and Synthesis of Sampled-Date Control Systems." Prentice-Hall, Inc., 1963.

3. Tou, J. T.: "Optimum Design of Digital Control Systems." Academic Press Inc., 1963.

APPENDIX I

PROGRAM PHIDEL

```
      PROGRAM PHIDEL
     0DIMENSION F(12,12),PHI(12,12),TERM(12,12),WORM(12,12),
     1DEL(12),DELM(12,12),TELM(12,12),DELP(12,12),D(12)

C  THIS PROGRAM CALCULATES THE STATE TRANSITION MATRIX, PHI, AND THE
C  EXTERNAL FORCING FUNCTIONS IMPULSE RESPONSE MATRIX, DEL, FROM THE
C  MATRICES, F AND D.  THIS IS ACCOMPLISHED BY USING THE TAYLOR SERIES
C  EXPANSION FOR THE EXPONENTIAL FUNCTION.  THE CALCULATION FOR THE
C  DEL MATRIX IS GOOD FOR ONLY NON TIME VARYING DYNAMICS BECAUSE THE
C  INTEGRATION IS ACCOMPLISHED WITHIN THE PROGRAM FOR THIS CASE ONLY.

C  PHI=EXP**(F*T)=I+(F*T)+((F*T)**2)/(1.0*2.0)+...+((F*T)**N)/(1.0*..*N)

C  DEL=INTEGRAL FROM ZERO TO DT OF PHI(DT-TAU)*D*DTAU

C  THE REQUIRED INPUT DATA CARDS ARE,

C  (A)  N (SIZE OF THE INPUT MATRIX, FORMAT I5)
C  (B)  DT (SAMPLING PERIOD, FORMAT 1F15.12)
C  (C)  TEST (TEST FOR MAXIMUM ALLOWABLE NUMBER IN A TERM TO TRUNCATE
C             THE SERIES, FORMAT 1F15.12)
C  (D)  F (DISTRIBUTION MATRIX, FORMAT MUST BE INSERTED, STATEMENT
C         NUMBER 31, NF10.6)
C  (E)  D (COLUMN VECTOR RELATING THE CONTROL VARIABLES TO THE SYSTEM
C           DYNAMICS, FORMAT 8F10.6)

C  OTHER FORMAT CARDS WHICH MUST BE INSERTED ARE NRS 34 AND 35 WHICH
```

```
C  ALLOWS PHI AND THE EXPONENTIAL SERIES TERMS TO BE PRINTED OUT IN
C  THEIR MATRIX FORM.  THIS PROGRAM TERMINATES WHEN ALL ELEMENTS OF THE
C  SERIES TERMS ARE LESS THAN THE TEST VARIABLE, THUS GUARANTEEING THAT
C  THE SUM OF ALL TERMS TO FOLLOW IS LESS THAN THE LAST ELEMENTS
C  CALCULATED.


C  READ DATA AND INITIALIZE


      READ 32 (N)
      READ 33 (DT)
      READ 33 (TEST)
      READ 31 ((F(IR,IC),IC=1,N),IR=1,N)
      READ 31 (D(I),I=1,N)
      TM=0.0
      PRINT 56,DT
      PRINT 54 ((F(IR,IC),IC=1,N),IR=1,N)
      PRINT 55 (D(I),I=1,N)
      DO 40 IR=1,N
      DO 40 IC=1,N
      TERM(IR,IC)=0.0
      WORM(IR,IC)=0.0
      TERM(IR,IR)=1.0
      TELM(IR,IC)=TERM(IR,IC)*DT
      DELP(IR,IC)=TELM(IR,IC)
      DELM(IR,IC)=0.0
      DEL(IR)=0.0
   40 PHI(IR,IC)=TERM(IR,IC)


C  CALCULATE THE ELEMENTS OF EACH TERM IN THE SERIES


   44 TM=1.0+TM
```

```
      DO 50 IR=1,N
      DO 50 IC=1,N
      DO 50 JN=1,N
      DELM(IR,IC)=DELM(IR,IC)-TELM(IR,JN     (JN,IC)*DT/(TM+1.0)
   50 WORM(IR,IC)=TERM(IR,JN)*F(JN,IC)*       +WORM(IR,IC)
      DO 41 IR=1,N
      DO 41 IC=1,N
      TERM(IR,IC)=WORM(IR,IC)
      TELM(IR,IC)=DELM(IR,IC)
      DELP(IR,IC)=DELP(IR,IC)+TELM(IR,IC)
      DELM(IR,IC)=0.0
   41 WORM(IR,IC)=0.0
      M=TM
      PRINT 34,M,((TERM(IR,IC),IC=1,N),IR=1,N)

C     SUMATION OF THE TERMS IN THE SERIES

      DO 51 IR=1,N
      DO 51 IC=1,N
   51 PHI(IR,IC)=PHI(IR,IC)+TERM(IR,IC)

C     TEST FOR THE TERMINAL TERM IN THE SERIES AND PRINT THE PHI MATRIX

      ABC=0.0
      DO 42 IR=1,N
      DO 42 IC=1,N
      AA=TERM(IR,IC)
      AB=ABSF(AA)
      IF(ABC-AB)43,43,42
   43 ABC=AB
   42 CONTINUE
```

```
      IF (TEST-ABC)45,45,46
   45 GO TO 44
   46 PRINT 35 ((PHI(IR,IC),IC=1,N),IR=1,N)

C     CALCULATE AND PRINT THE DEL MATRIX

      DO 52 I=1,N
      DO 52 K=1,N
      DO 52 J=1,N
   52 DEL(I)=DEL(I)+PHI(I,J)*DELP(J,K)*D(K)
      PRINT 53 (DEL(I),I=1,N)
   31 FORMAT (8F10.6)
   32 FORMAT (I5)
   33 FORMAT (1F15.12)
   34 FORMAT (///7HFT(I,J),I2/(3F12.8))
   35 FORMAT (////9X,8HPHI(I,J)///(3F12.8))
   53 FORMAT (////9X,6HDEL(I)//(8F15.9)//)
   54 FORMAT (//,7HF(I,J)=//,(3F10.4))
   55 FORMAT (//,5HD(I)=//,(8F10.4))
   56 FORMAT (//,3HDT=,(1F10.8))
      END
      END
```

APPENDIX II

PROGRAM OPCON1

```
      PROGRAM OPCON1
     0DIMENSION A(8,8),B(8,8),Q(8,8),D(8,8),X(8,40),G(40),H(8,8),
     1P(40,8,8),R(40,8,8),Z(40,8,8),COST(40),Y(8,8),ZZ(8,8),
     2RR(8,8),P1(8,8),SM(40,8,8),F(40,8,8),AA(8,8),XX(8,8),E(8,8),
     3MON(8,8),SMM(8,8),U(40),UU(8,8)

C   THIS PROGRAM IS DERIVED USING THE COST FUNCTION, J(N)=MINIMUM(SUM
C   X(N)T*Q*X(N)). THE PROGRAM WILL YIELD OPTIMUM CONTROL FOR UP TO AN
C   EIGHTH ORDER SYSTEM OVER 40 STAGES. THE INPUT DATA CARDS ARE,

C   (A) M (NUMBER OF STAGES PLUS ONE, FORMAT I5)
C   (B) N (ORDER OF SYSTEM, FORMAT I5)
C   (C) DEL (COLUMN MATRIX READ IN ORDER BY ROW, FORMAT 8F10.6)
C   (D) PHI (TRANSITION MATRIX READ IN A COLUMN AT A TIME, FORMAT 8F10.6)
C   (E) Q (NXN WEIGHTING MATRIX READ IN SAME AS PHI, FORMAT 8F10.6)
C   (F) X (COLUMN VECTOR OF INITIAL STATE CONDITIONS, FORMAT 8F10.6)

C   THE FOLLOWING RECURSIVE EQUATIONS WERE DERIVED AND ARE USED AS THE
C   BASIS FOR THE OPTIMIZATION OF CONTROL,

C   PSI(K+1)=(I-(DEL*DELT*(Q+P(K)))/(DELT*(Q+P(K))*DEL))*PHI          (1)

C   P(K-1)=PSIT(K-1)*P(K-2)*PSI(K-1)                                  (2)

C   U(N-K)=-((DELT*(Q+P(K-1))*PHI)/)DELT*(Q+P(K-1))*DEL))*X(N-K)      (3)

C   X(K+1)=PHI*X(K)+DEL*U(K)                                          (4)
```

```
C  EQUATIONS 1 AND 2 CONSTITUTE SUBROUTINES IN THIS PROGRAM. ANOTHER
C  SUBROUTINE GENERATES THE RESULTING COEFFICIENT MATRIX FOR X(N-K) IN
C  EQUATION 3. (SUBROUTINE FF)

      READ 202 (M)
      READ 202 (N)
      READ 201 (A(I,1),I=1,N)
      READ 201 ((B(I,J), I=1,N),J=1,N)
      READ 201 ((Q(I,J),I=1,N),J=1,N)
      READ 201 (X(I,1),I=1,N)
      PRINT 202 (M)
      PRINT 202 (N)
      PRINT 201 (A(I,1),I=1,N)
      PRINT 201 ((B(I,J),I=1,N),J=1,N)
      PRINT 201 ((Q(I,J),I=1,N),J=1,N)
      PRINT 201 (X(I,1),I=1,N)
      CALL DELTA (D,A,N)
      DO 71 I=1,N
      DO 71 J=1,N
      Z(1,I,J)=Q(I,J)
   71 ZZ(I,J)=Z(1,I,J)
      DO 72 I=1,M-1
      CALL RPSI(D,ZZ,A,B,RR,N)
      CALL PP (RR,Q,P1,N)
      CALL SUM(ZZ,Q,P1,N)
      DO 72 J=1,N
      DO 72 K=1,N
      P(I,J,K)=P1(J,K)
      R(I,J,K)=RR(J,K)
   72 Z(I+1,J,K)=ZZ(J,K)
      PRINT 1004
      PRINT 102 (((I,J,K,R(I,J,K),K=1,N),J=1,N),I=1,M)
      PRINT 1005
      PRINT 102 (((I,J,K,P(I,J,K),K=1,N),J=1,N),I=1,M)
```

```
      PRINT 1006
      PRINT 102 (((I,J,K,Z(I,J,K),K=1,N),J=1,N),I=1,M)
      DO 73 I=1,M
      DO 73 J=1,N
      DO 73 K=1,N
   73 SM(I,J,K)=Z(M-I,J,K)
      PRINT 1007
      PRINT 102 (((I,J,K,SM(I,J,K),K=1,N),J=1,N),I=1,M)
      PRINT 1002
      DO 74 I=1,M-1
      DO 75 J=1,N
      DO 75 K=1,N
   75 SMM(J,K)=SM(I,J,K)
      CALL FF(H,A,B,SMM,N)
      DO 20 J=1,N
      DO 20 K=1,N
   20 H(J,K)=-H(J,K)
      DO 76 J=1,N
      DO 76 K=1,N
   76 Y(J,K)=0.0
      DO 79 J=1,N
   79 Y(J,1)=X(J,I)
      CALL PROD (UU,H,Y,1,1,N)
      U(I)=UU(1,1)
      CALL PROD(AA,A,UU,N,1,1)
      CALL PROD (XX,B,Y,N,1,N)
      DO 77 J=1,N
   77 X(J,I+1)=XX(J,1)+AA(J,1)
      DO 78 J=1,N
      F(I,1,J)=H(1,J)
   78 E(J,1)=X(J,I)
      CALL MONEY(MON,E,Q,N)
      PRINT 100 (I,MON(1,1))
   74 COST(I)=MON(1,1)
      U(M)=0.0
```

APPENDIX II II-3 PROGRAM OPCON1 (CONTINUED)

```
      DO 85 I=1,N
      DO 85 J=1,N
   85 F(M,I,J)=0.0
      DO 86 I=1,N
   86 E(I,1)=X(I,M)
      CALL MONEY(MON,E,Q,N)
      PRINT 100 (M,MON(1,1))
      COST(M)=MON(1,1)
      PRINT 1000
      PRINT 100 (I,U(I),I=1,M)
      PRINT 1001
      PRINT 101 ((J,I,X(J,I),J=1,N),I=1,M)
      PRINT 1003
      PRINT 101 ((J,I,F(I,1,J),J=1,N),I=1,M)
  100 FORMAT (I5,F20.15)
  101 FORMAT (2I5,F20.15)
  102 FORMAT (3I5,F20.15)
  201 FORMAT(8F10.6)
  202 FORMAT(I5)
 1000 FORMAT (//,25H  M          U(M)          ,//)
 1001 FORMAT (//,30H  N     M          X(N,M)        ,//)
 1002 FORMAT (//,25H  M         COST(M)        ,//)
 1003 FORMAT (//,30H  N     M        F(M,1,N)        ,//)
 1004 FORMAT (//,35H  M   ROW  COL     PSI(M,ROW,COL)  ,//)
 1005 FORMAT (//,35H  M   ROW  COL       P(M,ROW,COL)  ,//)
 1006 FORMAT (//,35H  M   ROW  COL     Q+P(M,ROW,COL)  ,//)
 1007 FORMAT (//,35H  M   ROW  COL      SM(M,ROW,COL)  ,//)
      END

C  THE FOLLOWING SUBROUTINE CALCULATES PSI(N).

      SUBROUTINE PPSI  (B,C,D,E,F,N)
     0DIMENSION A(8,8),B(8,8),C(8,8),D(8,8),E(8,8),F(8,8),G(8,8),H(8,8),
     1AA(8,8),BB(8,8),DD(8,8),CC(8,8),EE(8,8)
      CALL TRANSQ (B,G,N)
```

```
      CALL PROD (AA,G,C,N,N,N)
      CALL PROD (BB,B,AA,N,N,N)
      CALL TRANCOL(D,DD,N)
      CALL PROD (CC,C,D,N,1,N)
      CALL PROD (EE,DD,CC,1,1,N)
      DO 60 I=1,N
      DO 60 J=1,N
   60 A(I,J)=0.0
      DO 61 I=1,N
   61 A(I,I)=1.0
      DO 62 I=1,N
      DO 62 J=1,N
   62 BB(I,J)=BB(I,J)/EE(1,1)
      DO 63 I=1,N
      DO 63 J=1,N
   63 BB(I,J)=A(I,J)-BB(I,J)
      CALL PROD (F,BB,E,N,N,N)
      END

C  THE FOLLOWING SUBROUTINE CALCULATES P.

      SUBROUTINE PP(A,B,C,N)
      DIMENSION A(8,8),B(8,8),C(8,8),D(8,8),AA(8,8)
      CALL TRANSQ(A,AA,N)
      CALL PROD(D,B,A,N,N,N)
      CALL PROD(C,AA,D,N,N,N)
      END

C  THE FOLLOWING SUBROUTINE CALCULATES THE COEF. OF THE STATE VARIABLES
C  FOR THE CONTROL POLICY.

      SUBROUTINE FF(A,B,C,D,N)
     0DIMENSION A(8,8),B(8,8),C(8,8),D(8,8),E(8,8),BB(8,8),G(8,8),
     1H(8,8)
      CALL PROD(E,D,C,N.N,N)
```

```
      CALL TRANCOL(B,BB,N)
      CALL PROD(A,BB,E,1,N,N)
      CALL PROD(G,D,B,N,1,N)
      CALL PROD(H,BB,G,1,1,N)
      DO 64 I=1,N
   64 A(1,I)=A(1,I)/H(1,1)
      END

C  THIS SUBROUTINE CALCULATES THE COST AT EACH STAGE.

      SUBROUTINE MONEY(MON,E,Q,N)
      DIMENSION MON(8,8),E(8,8),ET(8,8),Q(8,8),MO(8,8)
      CALL TRANCOL(E,ET,N)
      CALL PROD(MO,Q,E,N,1,N)
      CALL.PROD(MON,ET,MO,1,1,N)
      END

C  THIS SUBROUTINE TRANSPOSES ANY COLUME MATRIX (1,N)

      SUBROUTINE TRANCOL (A,B,N)
      DIMENSION A(8,8),B(8,8)
      DO 12 I=1,N
   12 B(1,I)=A(I,1)
      END

C  THIS SUBROUTINE TRANSPOSES ANY SQUARE MATRIX(NXN)

      SUBROUTINE TRANSQ(A,B,N)
      DIMENSION A(8,8),B(8,8)
      DO 11 I=1,N
      DO 11 J=1,N
   11 B(J,I)=A(I,J)
      END

C  THIS SUBROUTINE CALCULATES THE PRODUCT OF ANY TWO MATRICES AS LONG
```

```
C  AS THE PRODUCT IS ALGEBRAICALLY PROPER

      SUBROUTINE PROD (A,B,C,L,M,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 10 I=1,L
      DO 10 J=1,M
      A(I,J)=0.0
      DO 10 K=1,N
   10 A(I,J)=A(I,J)+B(I,K)*C(K,J)
      END
      SUBROUTINE DELTA(D,A,N)
      DIMENSION D(8,8),A(8,8)
      DO 42 I=1,8
      DO 42 J=1,8
   42 D(I,J)=0.0
      DO 41 I=1,N
   41 D(I,1)=A(I,1)
      END

C  THE FOLLOWING SUBROUTINE SUMS ANY TWO NXN MATRICES.

      SUBROUTINE SUM(A,B,C,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 51 I=1,N
      DO 51 J=1,N
   51 A(I,J)=B(I,J)+C(I,J)
      END
      END
```

APPENDIX III

PROGRAM OPCON2

```
      PROGRAM OPCON2
     0DIMENSION AT(8,8),ALFAT(2000,8),Q(8,8),P(8,8),P1(8,8),DEL(8,8),
     1PSI(8,8),PHI(8,8),X(8,8),Y(8,8),DELTA(8,8),U(8,8)

C   THIS PROGRAM HAS BEEN DERIVED USING A COST FUNCTION, J(N)=MINIMUM(SUM
C   X(N)T*Q*X(N)+SUM R*U(N-1)**2). THE DIMENSION STATEMENTS LIMIT THE
C   PROGRAM TO AN EIGHTH ORDER SYSTEM WITH 2000 ITERATIVE STAGES. THE
C   FOLLOWING ITERATIVE EQUATIONS WERE DEVELOPED AND DERIVED USING R.E.
C   BELLMANS PRINCIPLES OF DYNAMIC PROGRAMMING,

C   A(K)T=-(DELT*P(K-1)*PHI)/(DELT*P(K-1)*DEL+R)                    (1)

C   PSI(K)=PHI+DEL*A(K)T,     PSI(0)=0                               (2)

C   P(K)=PSI(K)T*P(K-1)*PSI(K)+Q+R*A(K)*A(K)T,    P(0)=Q            (3)

C   X(K)=PHI*X(K-1)+DEL*U(K-1)                                       (4)

C   U(N-K)=A(K)T*X(N-K)                                              (5)

C   EQUATIONS 1, 2, AND 3 CONSTITUTE SUBROUTINES IN THIS PROGRAM.

C   THE DATA CARDS REQUIRED ARE LISTED IN ORDER,

C   (A)  M  (NUMBER OF STAGES, FORMAT I5)
C   (B)  N  (ORDER OF SYSTEM, FORMAT I5)
C   (C)  IT  (MULTIPLE OF RESULTS OF STAGES TO BE PRINTED, FORMAT I5)
```

```
C  (D)  R  (CONSTANT IN THF COST FUNCTION WHICH WEIGHTS THE SUM OF THF
C           CONTROL, FORMAT 1F10.6)
C  (E)  DEL  (THE SYSTEM IMPULSE RESPONSE MATRIX, FORMAT 8F10.6, READ
C            IN BY COLUMN)
C  (F)  PHI  (THE SYSTEM TRANSITION MATRIX, FORMAT NF10.6, READ IN BY
C            ROWS)
C  (G)  Q  (WEIGHTING MATRIX USED IN THE COST FUNCTION, FORMAT NF10.6,
C           READ IN BY ROWS)
C  (H)  X  (INITIAL CONDITIONS ON THE COLUMN VECTOR OF STATES, FORMAT
C           8F10.6, READ IN BY COLUMN)

C  FORMAT STATEMENT NR 8 WILL HAVE TO BE ALTERED DEPENDING UPON THE
C  ORDER OF THE SYSTFM, ITS FORMAT SHOULD BE NF10.6.


C  READ DATA AND INITIALIZE VARIABLES.

      READ 2(M)
      READ 2 (N)
      READ 2 (IT)
      READ 3 (R)
      READ 1 (DEL(I,1),I=1,N)
      READ 8 ((PHI(I,J),J=1,N),I=1,N)
      RFAD 8 ((Q(I,J),J=1,N),I=1,N)
      READ 1 (X(I,1),I=1,N)
      PRINT 2 (M)
      PRINT 2 (N)
      PRINT 2 (IT)
      PRINT 3 (R)
      PRINT 1 (DEL(I,1),I=1,N)
      PRINT 8 ((PHI(I,J),J=1,N),I=1,N)
      PRINT 8 ((Q(I,J),J=1,N),I=1,N)
      PRINT 1 (X(I,1),I=1,N)
      TCOST=0.0
      KN=0
```

```
      DO 20 I=1,N
      DO 20 J=1,N
   20 P1(I,J)=Q(I,J)

C     CALCULATE AND PRINT A(K)T, PSI(K), AND P(K).

      DO 21 I=1,M-1
      CALL ATRAN(AT,P1,PHI,DEL,R,N)
      DO 22 J=1,N
   22 ALFAT(M-I,J)=AT(1,J)
      CALL PPSI(PSI,PHI,DEL,AT,N)
      CALL PP(P,PSI,P1,Q,AT,R,N)
      KN=KN+1
      IF (KN-IT)17,18,18
   18 PRINT 4,I,(AT(1,J),J=1,N)
      PRINT 5,I,((PSI(J,K),K=1,N),J=1,N)
      PRINT 6,I,((P(J,K),K=1,N),J=1,N)
      KN=0
   17 CONTINUE
      DO 21 J=1,N
      DO 21 K=1,N
   21 P1(J,K)=P(J,K)

C     CALCULATE AND PRINT U(K), X(K), AND COST(K).

      DO 23 I=1,N
   23 AT(1,I)=ALFAT(1,I)
      DO 28 J=2,N
      DO 28 K=1,N
   28 AT(J,K)=0.0
      CALL PROD(U,AT,X,1,1,N)
      Z=U(1,1)
      CALL COST (DOL,X,Q,R,Z,N)
      TCOST=TCOST+DOL
      L=1
```

APPENDIX III   III-3   PROGRAM OPCON2 (CONTINUED)

```
      PRINT 7,L,Z,(X(I,1),I=1,N)
      PRINT 9 (TCOST)
      KN=1
      DO 24 I=2,M
      DO 25 J=1,N
   25 Y(J,1)=X(J,1)
      CALL PROD(X,PHI,Y,N,1,N)
      DO 26 J=1,N
      DELTA(J,1)=Z*DEL(J,1)
      X(J,1)=X(J,1)+DELTA(J,1)
   26 AT(1,J)=ALFAT(I,J)
      DO 29 J=2,N
      DO 29 K=1,N
   29 AT(J,K)=0.0
      CALL PROD(U,AT,X,1,1,N)
      Z=U(1,1)
      CALL COST (DOL,X,Q,R,Z,N)
      TCOST=TCOST+DOL
      KN=KN+1
      IF (KN-IT)24,27,27
   27 PRINT 7,I,Z,(X(I,1),I=1,N)
      PRINT 9 (TCOST)
      KN=0
   24 CONTINUE
    1 FORMAT (8F10.6)
    2 FORMAT (I5)
    3 FORMAT (1F10.6)
    4 FORMAT (//,2HM=,I5,/,2HAT,8X,8F10.6,//)
    5 FORMAT (//,2HM=,I5,/,3HPSI,7X,8F10.6,//)
    6 FORMAT (//,2HM=,I5,/,1HP,9X,8F10.6,//)
    7 FORMAT (//,2HM=,I5,3X,2HU=,1F10.6,/,10X,2HX=,8F10.6,//)
    8 FORMAT (3F10.6)
    9 FORMAT (30X,6HTCOST=,1F15.6)
      END
```

```
C   THIS SUBROUTINE CALCULATES A(K)T.

      SUBROUTINE ATRAN (AT,P,PHI,DEL,R,N)
     0DIMENSION AT(8,8),P(8,8),PHI(8,8),DEL(8,8),DELT(8,8),AB(8,8),
     1AC(8,8),AD(8,8)
      CALL TRANCOL (DEL,DELT,N)
      CALL PROD(AB,DELT,P,1,N,N)
      CALL PROD(AC,[illegible],DEL,1,1,N)
      CALL PROD(AD,AB,PHI,1,N,N)
      DO 14 I=1,N
   14 AT(1,I)=-AD(1,I)/(AC(1,1)+R)
      END

C   THIS SUBROUTINE CALCULATES PSI(K).

      SUBROUTINE PPSI(PSI,PHI,DEL,AT,N)
      DIMENSION PSI(8,8),PHI(8,8),DEL(8,8),AT(8,8),AB(8,8)
      CALL PROD (AB,DEL,AT,N,N,1)
      CALL SUM (PSI,PHI,AB,N)
      END

C   THIS SUBROUTINE CALCULATES P(K).

      SUBROUTINE PP(P,PSI,P1,Q,AT,R,N)
     0DIMENSION P(8,8),P1(8,8),PSI(8,8),Q(8,8),AT(8,8),AA(8,8),
     1PSIT(8,8),AC(8,8),AD(8,8),AE(8,8)
      DO 15 I=1,N
   15 AA(I,1)=AT(1,I)
      CALL TRANSQ (PSI,PSIT,N)
      CALL PROD (AB,PSIT,P1,N,N,N)
      CALL PROD(AC,AB,PSI,N,N,N)
      CALL PROD(AD,AA,AT,N,N,1)
      DO 16 I=1,N
      DO 16 J=1,N
   16 AD(I,J)=R*AD(I,J)
```

```
      CALL SUM (AE,AC,Q,N)
      CALL SUM(P,AE,AD,N)
      END

C   THIS SUBROUTINE TRANSPOSES A COLUMN MATRIX HAVING A MAXIMUM OF
C   EIGHT ELEMENTS.

      SUBROUTINE TRANCOL (A,B,N)
      DIMFNSION A(8,8),B(8,8)
      DO 12 I=1,N
   12 B(1,I)=A(I,1)
      END

C   THIS SUBROUTINE TRANSPOSES A SQUARE MATRIX OF MAXIMUM ORDER 8X8.

      SUBROUTINE TRANSQ (A,B,N)
      DIMENSION A(8,8),B(8,8)
      DO 11 I=1,N
      DO 11 J=1,N
   11 B(J,I)=A(I,J)
      FND

C   THIS SUBROUTINE CALCULATES THE COST AT EACH ITERATION AS DETERMINED
C   BY THE COST FUNCTION.

      SUBROUTINE COST (DOL,Y,Q,R,Z,N)
      DIMENSION Y(8,8),Q(8,8),Z(8,8),YT(8,8),A(8,8),AA(8,8)
      CALL TRANCOL(Y,YT,N)
      CALL PROD(A,YT,Q,1,N,N)
      CALL PROD(AA,A,Y,1,1,N)
      DOL=AA(1,1)+R*Z*Z
      END

C   THIS SUBROUTINE MULTIPLIES ANY TWO MATRICES WHICH ARE LIMITED TO
C   TWO 8X8S. THE ARGUMENTS ARE DEFINED AS FOLLOWS,
```

```
C   (A)  A   THE PRODUCT
C   (B)  B   THE MULTIPLICAND
C   (C)  C   THE MULTIPLIFR
C   (D)  L   NUMBER OF ROWS OF THE MULTIPLICAND AND PRODUCT
C   (E)  M   NUMBER OF COLUMNS OF THE MULTIPLIER AND PRODUCT
C   (F)  N   NUMBER OF COLUMNS OF THE MULTIPLICAND AND THE NUMBER OF ROWS
C            OF THE MULTIPLIER

      SUBROUTINE PROD (A,B,C,L,M,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 10 I=1,L
      DO 10 J=1,M
      A(I,J)=0.0
      DO 10 K=1,N
   10 A(I,J)=A(I,J)+B(I,K)*C(K,J)
      END

C   THIS SUBROUTINE FINDS THE SUM OF ANY TWO SQUARE MATRICES OF THE SAME
C   DIMENSIONS UP TO AN 8X8.

      SUBROUTINE SUM (A,B,C,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 13 I=1,N
      DO 13 J=1,N
   13 A(I,J)=B(I,J)+C(I,J)
      END
      FND
```

APPENDIX IV

PROGRAM OPCON3

```
      PROGRAM OPCON3
      DIMENSION AT(8,8),Q(8,8),P(8,8),P1(8,8),DEL(8,8),PSI(8,8),
     1PHI(8,8)

C   THIS PROGRAM UTILIZES A COST FUNCTION. J(N)=MINIMUM(SUM X(N)T*Q*X(N)+
C   SUM R*U(N-1)**2).  AN UNLIMITED NUMBER OF ITERATIONS MAY BE MADE AT
C   A COMPUTATION RATE OF 2000 PER MINUTE AFTER THE PROGRAM HAS BEEN
C   COMPILED.  THE OUTPUT OF THIS PROGRAM IS THE FEEDBACK GAIN MATRIX,
C   A TRANSPOSE.  THE FOLLOWING RECURSIVE EQUATIONS WERE DERIVED USING
C   DYNAMIC PROGRAMMING,

C   A(K)T=-(DELT*P(K-1)*PHI)/(DELT*P(K-1)*DEL+R)                (1)

C   PSI(K)=PHI+DEL*A(K)T,     PSI(0)=0                          (2)

C   P(K)=PSI(K)T*P(K-1)*PSI(K)+Q+R*A(K)*A(K)T,    P(0)=Q        (3)


C   EQUATIONS 1, 2, AND 3 CONSTITUTE SUBROUTINES IN THIS PROGRAM.

C   THE DATA CARDS REQUIRED ARE LISTED IN ORDER,

C   (A)  M   (NUMBER OF STAGES, FORMAT I5)
C   (B)  N   (ORDER OF SYSTEM, FORMAT I5)
C   (C)  IT  (MULTIPLE OF RESULTS OF STAGES TO BE PRINTED, FORMAT I5)
C   (D)  R   (CONSTANT IN THE COST FUNCTION WHICH WEIGHTS THE SUM OF THE
C             CONTROL, FORMAT 1F10.6)
```

```
C   (E)  DEL  (THE SYSTEM IMPULSE RESPONSE MATRIX, FORMAT 8F10.6, READ
C              IN BY COLUMN)
C   (F)  PHI  (THE SYSTEM TRANSITION MATRIX, FORMAT NF10.6, READ IN BY
C              ROWS)
C   (G)  Q  (WEIGHTING MATRIX USED IN THE COST FUNCTION, FORMAT NF10.6,
C             READ IN BY ROWS)

C   FORMAT STATEMENT NR 8 WILL HAVE TO BE ALTERED DEPENDING UPON THE
C   ORDER OF THE SYSTEM. ITS FORMAT SHOULD BE NF10.6.


C   READ DATA AND INITIALIZE VARIABLES.

      READ 2(M)
      READ 2 (N)
      READ 2 (IT)
      READ 3 (R)
      READ 1 (DEL(I,1),I=1,N)
      READ 8 ((PHI(I,J),J=1,N),I=1,N)
      READ 8 ((Q(I,J),J=1,N),I=1,N)
      PRINT 2 (M)
      PRINT 2 (N)
      PRINT 2 (IT)
      PRINT 3 (R)
      PRINT 1 (DEL(I,1),I=1,N)
      PRINT 8 ((PHI(I,J),J=1,N),I=1,N)
      PRINT 8 ((Q(I,J),J=1,N),I=1,N)
      KN=0
      DO 20 I=1,N
      DO 20 J=1,N
   20 P1(I,J)=Q(I,J)

C   CALCULATE AND PRINT A(K)T, PSI(K), AND P(K).

      DO 21 I=1,M-1
```

```
      CALL ATRAN(AT,P1,PHI,DEL,R,N)
      CALL PPSI(PSI,PHI,DEL,AT,N)
      CALL PP(P,PSI,P1,Q,AT,R,N)
      KN=KN+1
      IF (KN-IT)17,18,18
   18 PRINT 4,I,(AT(1,J),J=1,N)
      PRINT 5,I,((PSI(J,K),K=1,N),J=1,N)
      PRINT 6,I,((P(J,K),K=1,N),J=1,N)
      KN=0
   17 CONTINUE
      DO 21 J=1,N
      DO 21 K=1,N
   21 P1(J,K)=P(J,K)
    1 FORMAT (4F15.10)
    2 FORMAT (I5)
    3 FORMAT (1F10.6)
    4 FORMAT (//,2HM=,I2,/,2HAT,8X,8F10.6,//)
    5 FORMAT (//,2HM=,I2,/,3HPSI,7X,8F10.6,//)
    6 FORMAT (//,2HM=,I2,/,1HP,9X,8F10.6,//)
    8 FORMAT (2F10.6)
      END

C  THIS SUBROUTINE CALCULATES A(K)T.

      SUBROUTINE ATRAN (AT,P,PHI,DEL,R,N)
     0DIMENSION AT(8,8),P(8,8),PHI(8,8),DEL(8,8),DELT(8,8),AB(8,8),
     1AC(8,8),AD(8,8)
      CALL TRANCOL (DEL,DELT,N)
      CALL PROD(AB,DELT,P,1,N,N)
      CALL PROD(AC,AB,DEL,1,1,N)
      CALL PROD(AD,AB,PHI,1,N,N)
      DO 14 I=1,N
   14 AT(1,I)=-AD(1,I)/(AC(1,1)+R)
      END
```

```
C   THIS SUBROUTINE CALCULATES PSI(K).

       SUBROUTINE PPSI(PSI,PHI,DEL,AT,N)
       DIMENSION PSI(8,8),PHI(8,8),DEL(8,8),AT(8,8),AB(8,8)
       CALL PROD (AB,DEL,AT,N,N,1)
       CALL SUM (PSI,PHI,AB,N)
       END

C   THIS SUBROUTINE CALCULATES P(K).

       SUBROUTINE PP(P,PSI,P1,Q,AT,R,N)
      0DIMENSION P(8,8),P1(8,8),PSI(8,8),Q(8,8),AT(8,8),AA(8,8),
      1PSIT(8,8),AC(8,8),AD(8,8),AE(8,8)
       DO 15 I=1,N
    15 AA(I,1)=AT(1,I)
       CALL TRANSQ (PSI,PSIT,N)
       CALL PROD (AB,PSIT,P1,N,N,N)
       CALL PROD(AC,AB,PSI,N,N,N)
       CALL PROD(AD,AA,AT,N,N,1)
       DO 16 I=1,N
       DO 16 J=1,N
    16 AD(I,J)=R*AD(I,J)
       CALL SUM (AE,AC,Q,N)
       CALL SUM(P,AE,AD,N)
       END

C   THIS SUBROUTINE TRANSPOSES A COLUMN MATRIX HAVING A MAXIMUM OF
C   EIGHT ELEMENTS.

       SUBROUTINE TRANCOL (A,B,N)
       DIMENSION A(8,8),B(8,8)
       DO 12 I=1,N
    12 B(1,I)=A(I,1)
       END
```

```
C   THIS SUBROUTINE TRANSPOSES A SQUARE MATRIX OF MAXIMUM ORDER 8X8.

      SUBROUTINE TRANSQ (A,B,N)
      DIMENSION A(8,8),B(8,8)
      DO 11 I=1,N
      DO 11 J=1,N
   11 B(J,I)=A(I,J)
      END

C   THIS SUBROUTINE MULTIPLIES ANY TWO MATRICES WHICH ARE LIMITED TO
C   TWO 8X8S. THE ARGUMENTS ARE DEFINED AS FOLLOWS,

C   (A)  A   THE PRODUCT
C   (B)  B   THE MULTIPLICAND
C   (C)  C   THE MULTIPLIER
C   (D)  L   NUMBER OF ROWS OF THE MULTIPLICAND AND PRODUCT
C   (E)  M   NUMBER OF COLUMNS OF THE MULTIPLIER AND PRODUCT
C   (F)  N   NUMBER OF COLUMNS OF THE MULTIPLICAND AND THE NUMBER OF ROWS
C                 OF THE MULTIPLIER

      SUBROUTINE PROD (A,B,C,L,M,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 10 I=1,L
      DO 10 J=1,M
      A(I,J)=0.0
      DO 10 K=1,N
   10 A(I,J)=A(I,J)+B(I,K)*C(K,J)
      END

C   THIS SUBROUTINE FINDS THE SUM OF ANY TWO SQUARE MATRICES OF THE SAME
C   DIMENSIONS UP TO AN 8X8.

      SUBROUTINE SUM (A,B,C,N)
      DIMENSION A(8,8),B(8,8),C(8,8)
      DO 13 I=1,N
```

```
      DO 13 J=1,N
   13 A(I,J)=B(I,J)+C(I,J)
      END
      END
```

APPENDIX IV   IV-6   PROGRAM OPCON3 (CONCLUDED)